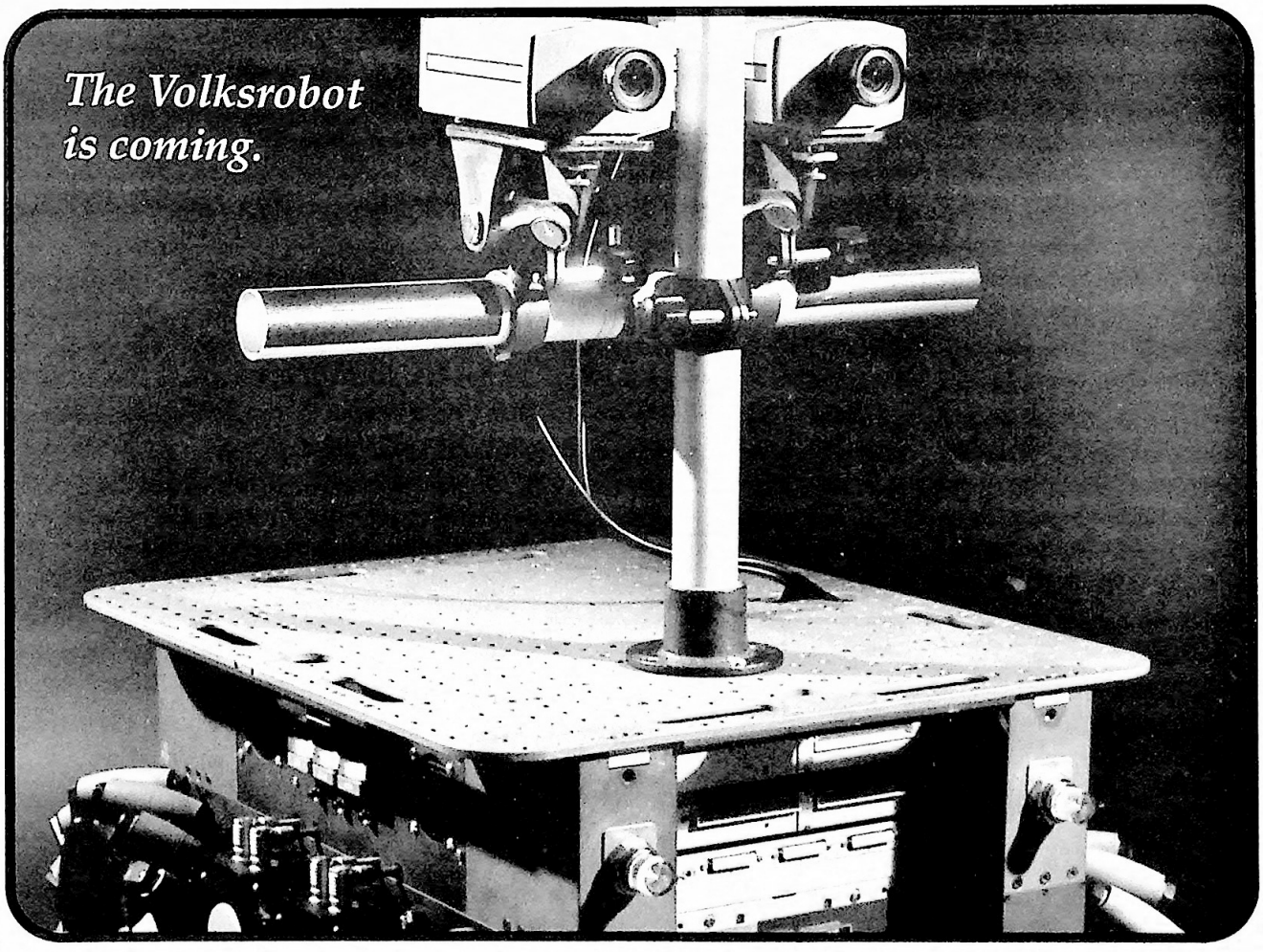
MARCH/ APRIL 1990

Algorithm

1.3

THE PERSONAL PROGRAMMING NEWSLETTER



Lightning Lines
Cellular Automata
Alphie and Omega

When you get your first issue of Amygdala you'll wonder how you ever did without it!

What's "Amygdala"?

Amygdala's a newsletter. It's devoted to fractals like the Mandelbrot set & Julia sets as well as other iteration sets.

What will I find in it?

- Fractal pictures;
- Articles about generating fractals on computers;
- Fractal bibliography;
- Reviews of fractal programs;
- Elementary & advanced articles about the math background underlying fractals.
- Et cetera...

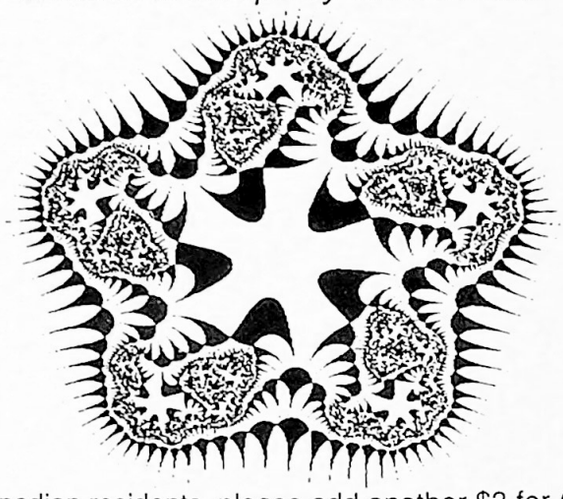
How can I get it?

- Just send $25 US for ten issues to:

  Amygdala
  Box 219
  San Cristobal, NM 87564 (USA)

Check, credit card, money order, cash...

A fractal biomorph by Ian Entwistle



Canadian residents, please add another $3 for Air Mail postage.

An additional $32 will get you the color slide supplement–forty incredible fractal slides. Each set of four 35 mm slides accompanies an issue of the newsletter.

EXPLOR CHAOS

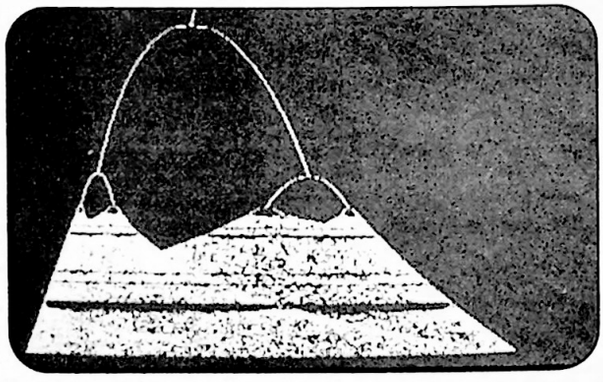
with

EXPLOR-I

from

TURING OMNIBUS

P.O. Box 1456,
London, Ontario
CANADA N6A 5M2



For PC and compatible computers with CGA compatible graphics.

EXPLOR-I is a package of programs designed for amateur and professional explorers alike. Select up to eight programs for your own personal package: Julia and Mandelbrot sets, Wallpaper for the Mind, Exploring Chaos, The Invisible Professor, The Cyclic Space Cellular Automaton, Star Clusters Simulation, Computer Popcorn, Biomorphs. In addition, we now offer the Core War Coliseum, the official Core War programming and tournament package. Write for our free catalogue to-day.

Algorithm

THE PERSONAL PROGRAMMING NEWSLETTER

ALGOLETTER - A. K. Dewdney
Of Newsletters and Space Stations 2

EASY PIECES - Michael W. Ecker
Lightning lines illuminate a world of graphics 4

PERSONAL PROGRAMS - Clifford A. Pickover
A computational tour of checkerboard worlds 8

ALGOPUZZLE - Dennis Shasha
Dr. Ecco demoralizes the matadors 12

ALGOFACT - Hans Moravec
One small step for a Volksrobot.....14

ALGOFICTION - Anthony Zweistein
Alphie and Omega - The Sequel.....18

BULLETIN BOARD
Software, games and publications available from individuals and small companies.....24

Algostaff

Editor
A.K. Dewdney

Assistant Editors
Jennifer Connor
John O'Hagan

Managing Editor
Louis D. Magguilli

Layout & Design
Tammy Golinski

Production Services
Two Oxford West

Address all editorial correspondence to A.K. Dewdney, 2 Oxford Street West, London, Ontario N6H 1P9, CANADA. Address advertising, business and all other correspondence to Louis D. Magguilli, Algorithm, P.O. Box 29237, Westmount Postal Outlet, 785 Wonderland Road S., London, Ontario N6K 1M6. Owned and published bimonthly by 854832 Ontario Inc., doing business as *Algorithm*.

All rights reserved, including the right to accept or reject all editorial and advertising matter. Unsolicited materials must be accompanied by self-addressed return envelopes and postage or dollar equivalent. No part of this newsletter may be reproduced without the written permission of *Algorithm*. Copyright© 1990 by *Algorithm*.

Mail subscriptions: $29.95 (U.S.) per year; single copies $5.00 (U.S.). Overseas add $4.00 (U.S.) for shipping. ISSN 0843-9753

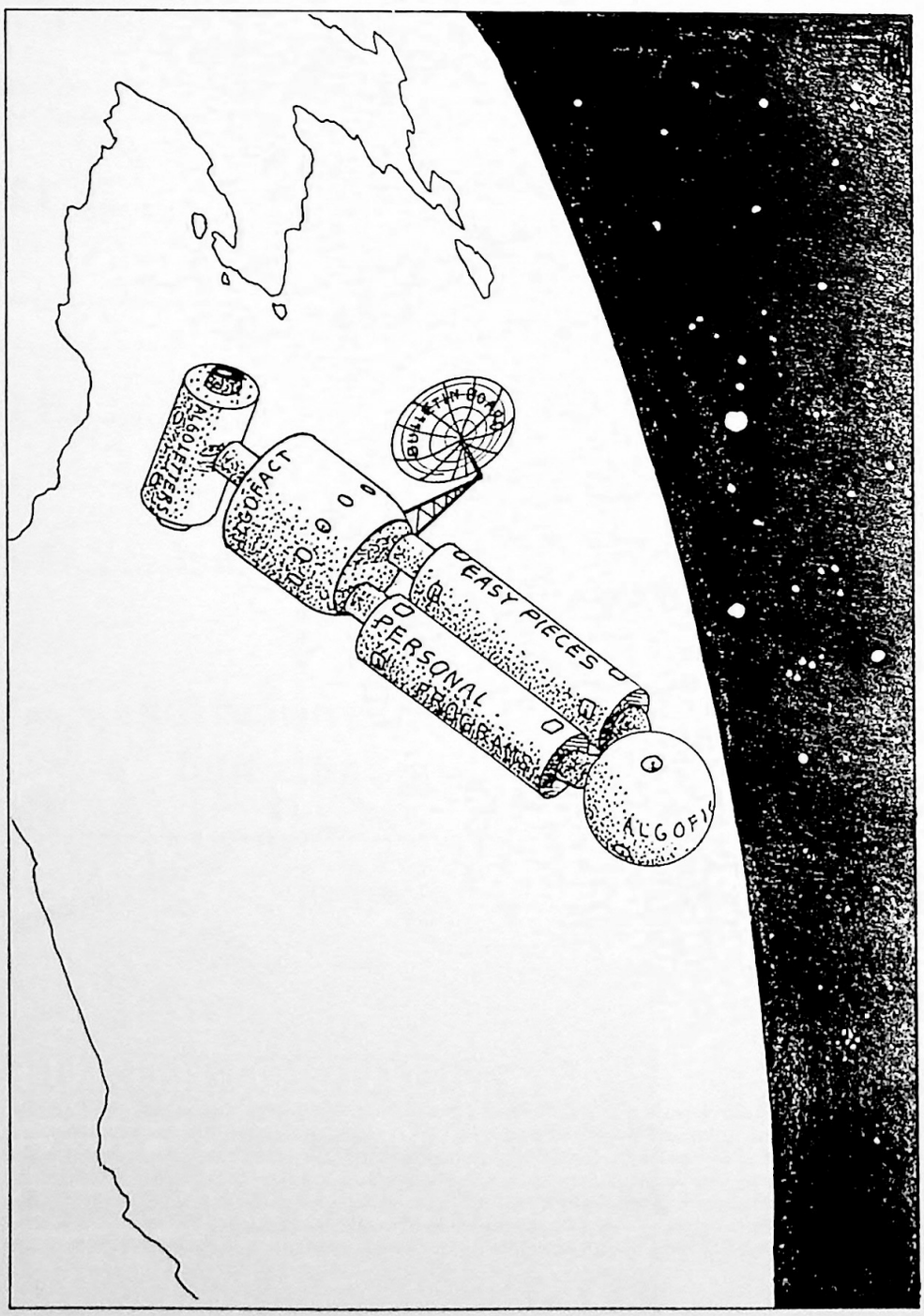
ALGOLETTER

by A.K. Dewdney

Of newsletters and space stations

Our newsletter reminds me of a space station. Its algorithmic emphasis carries it well above the atmosphere of particular languages and machines. In addition to this, it has a modular construction. We expect to loft up features, on occasion, to add to the growing structure, a place where more and more special interests find a home in weightless thought.

Indeed, our humble production headquarters sometimes remind me of a miniature launch facility, with costs to scale. Luckily, there are now enough paying customers to justify thoughts of expansion. It costs money (and time) to boost just one feature into orbit where it may be bolted to the current assembly.



TWO NEW FEATURES

The columns called Easy Pieces and Personal Programs fulfill a primary mission of our Enterprise: to launch both tyros and experts alike into new worlds of programming adventure, scientific exploration, and even aesthetic enjoyment. Currently, another feature is being readied on the ground, a new computer recreations column by the editor. It will nestle somewhat between Easy Pieces and Personal Programs in its emphasis. No more than middling difficulties should ever attend the many beautiful topics I am still privy to.

The ideas, mostly from readers, have been sitting in folders, quietly groaning for attention. Now, at last, I may usher some of them into the light of day.

Of course, many ideas will never be covered in a column that appears only six times a year. For this reason, we have designed a second feature to be launched at a later date. The new column will review experiments and explorations carried out by amateur and professional programmers alike.

Both features will be powered for a time by my backlog of "neat stuff." The right stuff, eventually, will be more current.

THE READER AS ASTRONAUT

Readers, as always, supply the right stuff by sending in descriptions of interesting adventures in computing: new cellular automata, cunning fractals, evolving systems, numerical magic, amazing patterns, experiments both real and imaginary. Based of such descriptions, we would attempt to build a column that condensed two or three into each issue of Algorithm. The work will be easier for descriptions upon which the author has already expended some care. Good writing is the next door neighbor of good programming.

ALGOLETTER OR ALGOLETTERS?

The last change calls for activation of a module currently in use but suffering from a lack of steady inhabitants. I refer to Algoletter, the vehicle of this very announcement. As an experiment of my own I now call on those creative souls with a few spare moments to describe their own project, program or predilection. These could begin to arrive in time for Algorithm 1.4. It is impossible to give a more precise idea of such Algoletters beyond giving examples. Any humor in the following titles is entirely accidental: "How I found the on/off switch on my new Toshiba Laptop," "A cellular automaton that imitates crab grass," "Using your computer screen as auxiliary memory," "A five-line program that induces hallucinations," "The most reader-friendly books on programming your PC in BASIC," "How to detect the Mother's Day Virus," and so on.

Letters should not exceed 200 words in length unless their contents happen to be mind-boggling.

THE BULLETIN BOARD

The Bulletin Board has been functioning quite well as a list of goods and services related to the personal programming adventure. We have posted notices for everything from hypercube programs to fractal calendars. The current policy on items acceptable for the Bulletin Board allows virtually anything that furthers the adventure — and more; since the first posting is always free, the Bulletin Board is an ideal vehicle not only for those with goods and services to sell or distribute, but for ideas and information as well.

Only imagination limits what may be posted in the latter category. For example, if a reader has produced a list of prime numbers, or has a little puzzle to propose, he or she may post a notice in the space provided on page 24 and 25 of this issue. Clubs and networks are especially welcome to advertise their advantages. Do you have a particularly nice idea that will fit in a Bulletin Board space? Publicize it!

There is nothing wrong with personal ads of a non-personal nature. For example, suppose you're a beginner who just can't seem to get your Mandelbrot program working. Explain your problem and appeal for assistance. Some kind soul might answer!

Finally, here is a slight but interesting modification to our Bulletin Board policy. It is practically guaranteed to enhance the visual potential of the medium: Whether you send us an interesting graphic for your Bulletin Board item, or whether we design one at your request, the items judged to have the greatest artistic merit will be left in a special pool. When we lay out Bulletin Board pages, we must work with multiples of ten items. Rather than omit freshly submitted items to make the numbers come out right, we will add old items — randomly drawn from the pool, of course. Such inclusions will be free.

We are determined not to raise the subscription price for ***Algorithm****. The more readers join us in orbit, the bigger and better the space station will get.*

Back issues of Algorithm

Now available, back issues fo Algorithm may be ordered at $5.00 (US) per copy.

Overseas orders please add $2.00 for postage and handling.

IMAGE CREDITS

Cover: Uranus robot courtesy of Dr. Hans Moravec, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213
Easy Pieces: photographs by Denise Saylor
Personal Programs: prints by Clifford A. Pickover
Algofact: Prints courtesy of Dr. Hans Moravec
Algoletter, Algopuzzle & Algofiction: drawings by A.K. Dewdney

Readings from
The Latter-day Cybernetician

"It shall not be known by an emergent species in a cellular or digital universe what its true substrate may be. For as much as it manifests with equal validity in silicon memory or gates of bamboo, the intelligence can in no wise determine, without external assistance, the computational support of its own being or that of its universe. It may attempt to enlist the cooperation of an external agent by forming signals within and without its own form. And these will,when formulated and carried out by one member of the species, be regarded as an aberration by its fellows."

EASY PIECES

by Michael W. Ecker

Lightning lines illuminate a world of graphics

Essential tools in any programmer's kit are the commands that draw basic shapes such as points, lines and circles on the computer screen. In this column we will play with the LINE command in BASIC. You may need a PC with a color graphics card and monitor in order for your system to recognize the commands. The higher the resolution of your screen (i.e., the density of points), the better, too.

Based on what we do here, you should be able to meet a special challenge that I will pose at the end of the column. In my next column in Algorithm 1.4, I will begin to answer some of the mail that has been piling up.

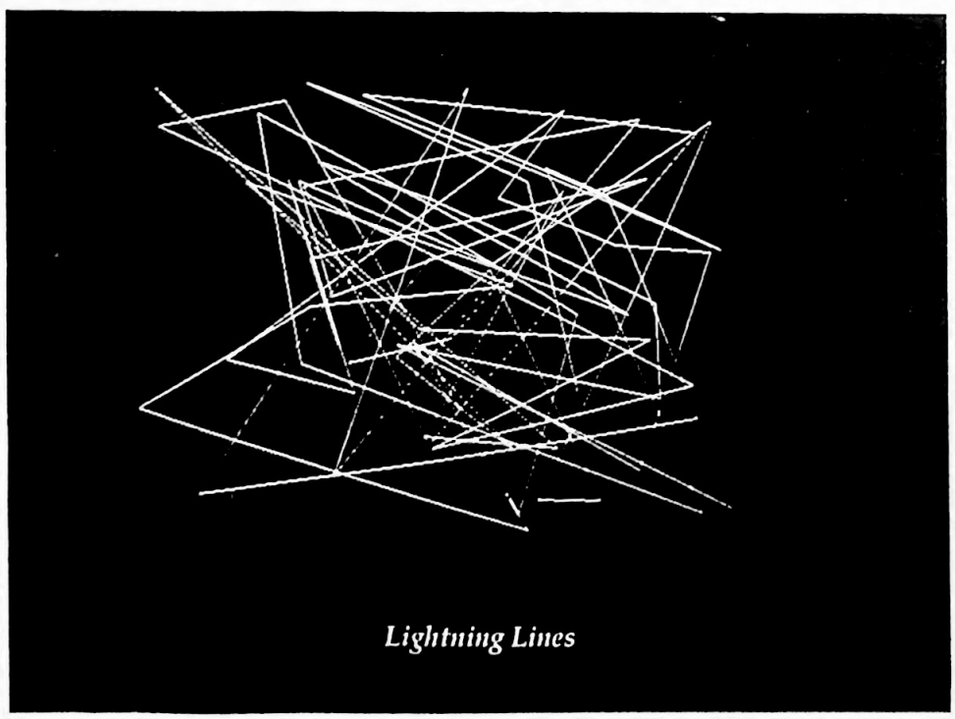
SCREENS, PIXELS AND COLOR

Your computer screen produces both text and graphics by employing the same basic technique —lighting individual dots, called pixels, on it. The more pixels your screen has in each square centimeter, the sharper an image your screen will produce, and the higher the resolution. If you have a color monitor, the screen has the additional ability to display any pixel in a variety of different colors. One widely available class of monitors is called RGB. It produces different colors by combining red (R), green (G) and blue (B) in different ways. Screens that light their pixels with just one color are called monochrome.

When they think of lines on computer screens, some readers may remember some of the older machines, like my ancient monochrome TRS-80 from Radio Shack. These screens were capable of only low-resolution graphics. In them, lines have jagged edges, looking like miniature staircases.

Of course, the higher the resolution, the less such lines would resemble staircases. If you remember any calculus, you might be reminded of how a curve is approximated by a succession of straight line segments. When the segments are long, the approximation is crude and the curve is ugly. When the segments are small, the approximation is much better and the curve looks smooth.

A high-resolution screen, likewise, will produce straight lines that look like straight lines and not staircases. Of the available types of color display, CGA (short for Color Graphics Adapter) has the lowest resolution, EGA (Enhanced Graphics Adapter) has a much higher resolution, and VGA (Video Graphics Array) has higher resolution still. In fact, while the EGA screen consists of 640 pixels across by 350 pixels down, the VGA screen has 640 pixels across and 480 down. VGA spin-offs that are adapted to desktop publishing, such as Super VGA, offer still higher resolution.



Lightning Lines

SOME BASIC GEOMETRY

There are essentially two different strategies for programming lines. One way involves lighting up a sequence of pixels by calculating their coordinates according to the dictates of a formula for a straight line. The other way is simpler. Merely use a built-in command for drawing a line all at once, so to speak. This command is available in GWBASIC and most other dialects of the language.

To make the excursion interesting, we will give ourselves some goals:

a) Use the line command to draw a square in the middle of the screen.

b) Use the line command to draw a dynamic display of lines that are drawn frenetically from point to point to point.

BASIC has several "screens" for displaying text or graphics. Ordinarily, when you enter BASIC, you are using screen 0 (by default). This screen is fine for alphanumeric characters but seems of little use when it comes to the graphing

commands that we'll be using.

If you enter the command SCREEN 1 by typing it, then touching the ENTER key, you will create a graphic mode that will support lines and other kinds of graphic figures. On my computers, the SCREEN 1 command never gives problems except in monochrome, where SCREEN 1 will not accept the LINE command. Entering SCREEN 1 creates a 40-column display. Personally, I dislike the appearance of text in a 40-column display. One can obtain the more normal 80-column display simply by typing WIDTH 80. Of course, either command, SCREEN 1 or WIDTH 80, can be typed by hand or included in a program as we shall see presently.

The horizontal dimension of a screen corresponds to the x-axis that we are all familiar with from high school. The vertical dimension corresponds to the y-axis with one little twist: down is up. In other words, the origin (0,0) is in the upper left corner of the screen, not the lower left. Thus, y-values increase as one goes down the screen.

For those readers who need refreshing, all points on the screen are given special coordinates, a pair of numbers. For example the point with coordinates (10,20) would lie 10 pixels to the right of the origin and 20 pixels below it. Ordinarily, we refer to "the point (10,20)" instead of "the point having coordinates (10,20)."

The screen, of course, is finite. The numbers allowed for coordinates of pixels have definite limits that depend on the resolution of the screen. On screens with good resolution, for example, the x-coordinates may range from 0 to 639 and the y-coordinates from 0 to, say, 199. In what follows I will frequently talk about "lines" when I really mean "line segment."

SOME LINE LORE

If you have your computer handy, you can start it up now and load BASIC in the usual way. If you don't have a computer handy, you can probably visualize yourself going through the following motions: Type the commands that set up screen 1 and width 80, then draw a line from (15,20) to (150,100). The facility for being able to type a few commands like this without creating a program is called "immediate mode." In this mode, you can fit all three commands on one line,

```
SCREEN 1: WIDTH 80: LINE (15,20)-(150,100)
```

and execute the commands simply by pressing the RETURN or ENTER key. By putting a line number in front of this command string, you will produce a miniature program that you can run by typing RUN:

```
10 SCREEN 1: WIDTH 80: LINE (15,20)-(150,100)
```

Either way, a line should appear on the screen with the usual OK prompt. Now change the numbers and experiment: Try typing LINE (0,100)-(100,0) or anything else you like. Note that the screen and width commands do not need to be repeated.

What about our first goal, to draw a square? If we agree to use 100 as the length of all four sides, then we might take (50,50), (50,150), (150,50) and (150,150) as the points at the four corners of the square. Since a square has four sides, you will need four LINE commands. What do they look like? Write a little program to do this. Those without computers can at least write out what the program should look like.

If you failed, you must carefully review your thinking process: For example, perhaps you joined the wrong pair of corners and got a diagonal instead of a side. Does your square look exactly like a square? If it seems close to a square but not quite so, you could be seeing some screen distortion, a subject I'll get into in a later column.

LINES.BAS:

```
10  KEY OFF: SCREEN 1: WIDTH 80
20  X1=0: Y1=0
30  X2=640*RND: Y2=200*RND
40  LINE (X1,Y1)-(X2,Y2)
50  X1=X2: Y1=Y2
60  GOTO 30
```

Meanwhile, concerning squares, I wonder whether any readers can think of some way of drawing squares without using the LINE command. How might you produce an octagon? Can you think of a good way to center figures that a program draws on the screen?

A LIGHTNING LINE PROGRAM

Our second goal called for "a dynamic display of lines that are drawn frenetically from point to point to point." What does this mean? Nothing less than producing a succession of lines from one randomly selected point to a second randomly selected point, then on to a third randomly selected point and so on, ad infinitum. Viewed as a cycle, the program must continually draw a line from the last point generated to a new one. This means that we must "swap" certain variables, as I'll explain in a moment. Here is an algorithm by which the program may be planned:

1. Take the first point at (0,0).
2. Take the second point by selecting x and y at random.
3. Draw a line from the first point to the second point.
4. Make the second point into the first point by swapping.
5. Go to step 2.

To "make the second point into the first point" is best explained if we have names for the coordinates

of the "first" and "second" points. Suppose the first point is (X1, Y1) and the second point is (X2,Y2). Then a program would make the second point into the first by moving the values currently in the coordinates X2 and Y2 into X1 and Y1. In the BASIC program in the box on page 5, this is done by two assignment statements on line 50: X1=X2 and Y1=Y2.

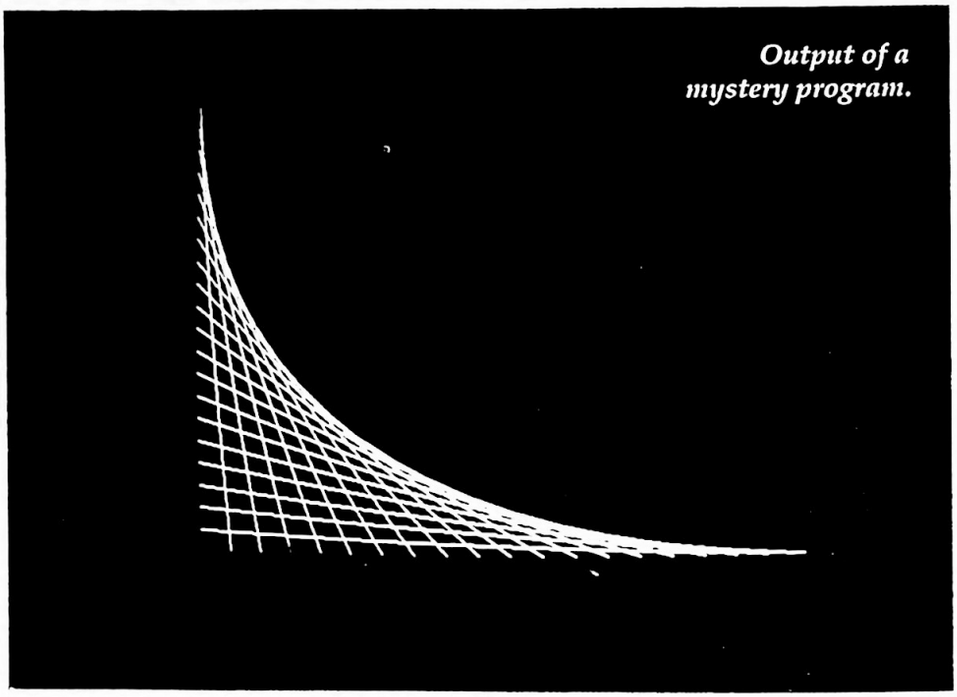
Notice that the program follows the algorithmic plan quite closely. Steps 1 through 5 of the algorithm match in a one-for-one manner with the program lines numbered 20 through 60, respectively. The program assumes that your screen happens to be 640 by 200. If you have EGA or VGA, it might be more pleasing to use screen 9 instead. Unfortunately, screen 9 is not supported by CGA.

Notice the use of colons to separate multiple commands on a single line. The KEY OFF command in line 10 simply turns off the system display at the bottom of the screen where the BASIC macro keys (F1: LIST, F2: RUN, etc.) are listed. The function keys continue to operate even though the list is erased. The program does not need a clear screen (CLS) command because SCREEN 1 does that automatically anyway.

In line 30, the program generates two random coordinates for the point (X2,Y2). It does this by using the RND (short for random) command. This produces a random number in decimal form between 0 and 1. Multiplying the number by the appropriate range limit, say 640, produces a new number between 0 and 640, just the thing for random x-coordinates. Multiplying by 200 produces a random candidate for y. Of course, if your screen has different dimensions from the ones assumed here, you must substitute these dimensions for 640 and 200 inside the instructions of line 30.

Line 40 corresponds to step 3 of the algorithm, where a line is to be drawn from the first point to the second point. The LINE command can take variables as arguments (what lies between the brackets) as well as particular numbers. It draws a line connecting the point (X1, Y1) to the point (X2, Y2), whatever the values of the variables X1, Y1, X2 and Y2 happen to be.

The first time line 40 is executed, (X1, Y1) happens to be the origin. Thereafter, X1 and Y1 are given the values previously in (X2, Y2). This happens in line 50 where what we might call a half-swap takes place. The instruction X1=X2 merely takes the number currently held in the variable X2 and places the same value in X1. The same with Y1 and Y2. In a full swap, the values previously in X1 and Y1 would be placed in X2 and Y2. (How might that be accomplished?)



Output of a mystery program.

LITNLINE.BAS

```
10   KEY OFF:  SCREEN 1:  WIDTH 80
20   X2=0:  Y2=0
30   X1=X2:  Y1=Y2
40   X2=640*RND:  Y2=200*RND
50   LINE (X1,Y1)-(X2,Y2)
60   COUNT=COUNT + 1
70   LOCATE 1,1:  PRINT COUNT
80   IF INKEY$<>""THEN GOSUB 100
90   GOTO 30
100  FOR DELAY=1 TO 10000:  NEXT
110  RETURN
120  REM:  repeated key presses will
130  REM:  slow the process down more
```

Finally, line 60 implements step 5 of the algorithm: the program's execution is sent back to line 30 where new random values for X2 and Y2 are computed. If you run this program you will see an endless succession of random lines appear on the screen, as in the illustration on page 4. Do not look at the display for long in case it induces a mild case of disorientation! The program stands in need of a few improvements, however.

For one thing, you might want to be able to stop the program, perhaps by the method described in Easy Pieces in Algorithm 1.1. Use the INKEY command to stop the program or, as in the jazzed-up program shown in the box on the left, slow it down by introducing a long delay before it starts up again. You may type this program in to see what it does. How does it work? Here is a good chance to do some detective work on your own. Look up any unfamiliar instructions in your manual or programming book.

Even a program that uses randomness can raise interesting mathematical puzzles. For example, how long should we expect the program LITNLINE to run, on average, before every pixel on the display rectangle is covered?

CLOSING THOUGHTS

Randomness is not everybody's cup of tea. For those who like more orderly graphics, there is a picture on the left produced by a program I call HYPRBL. My challenge to readers is to write and run a program that draws this picture. To start with, analyze the picture and write down an algorithm that will help you plan the program. Hint: each line in the figure joins a point of the form (Z,0) to a point of the form (0,Z).

Here's another, less visible challenge: Can you write a sequence of instructions that will perform a full swap of two variables, say A and B? If you have GWBASIC, let's agree not to use the built-in swap command. Your program must effect the swap by putting the value currently in A into B, without losing the value in B, and putting the original value of B in A. For example, if A currently equals 1 and B equals 0, your instructions should have the effect of putting 0 in A and 1 in B. In short, you cannot use any built-in swap command, nor can you use data values, nor can you use any foreknowledge of what values A and B might have.

If you would like me to look at your program for either challenge, please send it to the address below. You may send me the program(s) on paper or on MSDOS diskettes, preferably with text and/or program files saved in ASCII.

How would you like to create your own menu system so that you can call up any of the programs you write from a single list that is under your control? In my next column we will put that project on the menu.

Please send all questions, suggestions for future topics, and so on, to me at the following address:

Dr. Michael W. Ecker,
Easy Pieces/Algorithm
129 Carol Drive
Clarks Summit, PA 18411

If you wish me to acknowledge or to reply to your letter, please enclose a self-addressed and stamped (if possible) envelope, business-size or larger. Until Algorithm 1.4, may all your programs be Easy Pieces!

WHO READS Algorithm ?

systems managers, doctors, computer hobbyists, application programmers, high school students, lawyers, professors, hackers, airline pilots, engineers, researchers, teachers, stock brokers, university students, philosophers, reporters, entrepreneurs, able seamen, dreamers, technicians, game-designers, accountants, social workers, movers, shakers and candlestick makers.

WHO READS **Advertising**?

PERSONAL PROGRAMS

by Clifford A. Pickover

A computational tour of checkerboard worlds

He became aware of a kind of sparkle in the air ahead. Fairy lights blinking on and off. Cal saw three-dimensional patterns within the cloud, geometric ratios building and rebuilding in dazzling array....

OX,
Piers Anthony

So begins a science-fiction saga that describes humanity's first encounter with ephemeral entities resembling points of light on a three-dimensional checkerboard, and which move and change shape according to mathematical laws. Some readers will recognize the cloud as a three-dimensional cellular automaton. The theme saturates OX, even to the point where each chapter begins with a small cellular grid decorated by dots. The presence or absence of a dot in a grid cell would indicate which of two states a cell was in.

In general, a cellular automaton is an array of cells and a finite collection of possible states. At any given moment, each cell of the array must be in one of the allowed states. The rules that determine how the states of its cells change with time are what determine the cellular automaton's behavior. There is an infinite number of possible cellular automata, each like a checkerboard world in its own right. The world can be a one-dimensional strip of cells, a two-dimensional grid or, as we have seen above, a three-dimensional array. Of course, even higher dimensions are possible, but they are impossible to represent as clearly as the three lowest dimensions.

I will explore some personal favorites among the two-dimensional cellular automata and I will pass along some algorithms that you can feed your own computational vehicle, whether PC, Macintosh or mainframe.

Cellular automata comprise a class of simple mathematical systems which are fast becoming important as models for a variety of physical processes. Some cellular automata act in bizarre and random-looking ways while others exhibit highly ordered behavior. It all depends on the rules of the game. Cellular automata have been used to model the spread of plants, animals and even forest fires. They have mimicked fluid flow and chemical reactions. They have even been investigated as computers in their own right! Cellular automata are also referred to variously as "homogeneous structures," cellular structures" and "iterative arrays."

The concept of the cellular automaton was introduced in the 1950s by John von Neumann and Stanislaw Ulam. They saw in cellular automata an idealized system capable of modelling fundamental qualities of life itself. In fact, von Neumann succeeded in constructing a conceptual cellular automaton that was capable of replicating certain patterns of states in its cellular space. Self-reproduction seemed possible. By the 1960s, as computers became widespread in academic institutions, the Cambridge mathematician John Horton Conway became interested in cellular automata. Conway discovered a particular cellular automaton which he called Life, not only because its two states resembled life and death, but because computer experiments with certain configurations of cells produced behavior that could only be called life-like. The game was first publicized by Martin Gardner in his Mathematical Games column in the October 1970 issue of Scientific American. Since that time, cellular automata have become a very popular area of research for physicists, computer scientists and mathematicians.

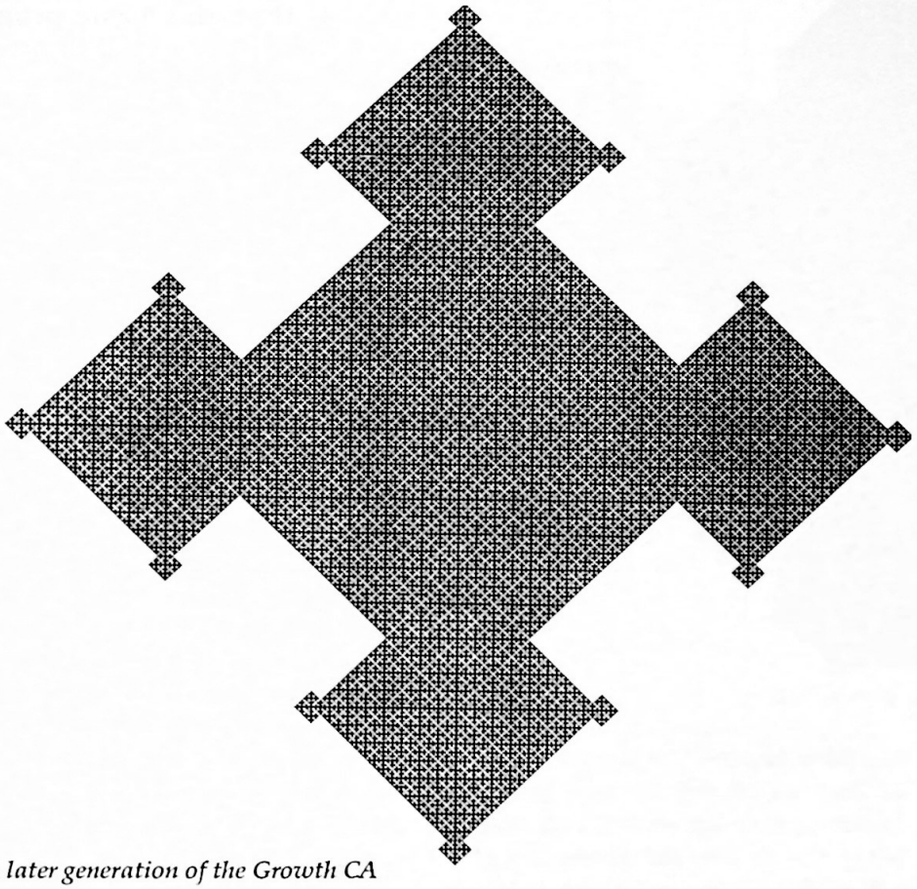
One reason for the potential of cellular automata as a research tool lies in the fact that any differential equation can be converted into a corresponding cellular automaton. This one simple fact opened the door to a brand new exploration of many differential equations, most of them being models for various physical processes of great interest to scientists.

The game of Life makes an ideal introduction to the subject of cellular automata. It is "played" on an infinite two-dimensional grid of cells, each cell being in one of two states (alive or dead) at any one time. At each tick of an abstract clock, each cell decides, in effect, whether it will be alive or dead at the next tick. All cells use exactly the same rules: Each cell considers its own state and the states of its eight neighbors, four along edges and another four at corners. The rules themselves are simple:

1. If a cell is alive at time t, it will remain alive at time t + 1 if it has have no more than 3 and no less than 2 living neighbors at time t.
2. If a cell is dead at time t, it will remain dead unless it has exactly 3 living neighbors. These act as parents.

Using these rules, Life can exhibit fantastically complicated and hard-to-predict behavior. The cellular game has spawned a publishing industry all its own. It is interesting that some scientists have suggested that, given a large enough array of cells in random states in a Life-like game, self-replicating entities would ultimately emerge, given enough time.

They might even evolve to



A later generation of the Growth CA

produce intelligent societies that develop and compete. It would be hard not to call such entities "alive."

CELLS THAT LIVE FOREVER

I will present a cellular automaton that differs somewhat from the game of Life. It also has configurations that grow according to certain rules but there the resemblance ends. I call it Growth, but I must credit the idea to Stanislaw Ulam, a cellular automaton pioneer. The rules of Growth dictate that any configuration will grow without limit as time progresses. The reason is simple. There is no rule for death. Every cell lives forever.

If I represent the two states of Growth by 0 and 1, the fate of a single 1, isolated amid 0s, is interesting to watch. In fact, you can simulate what happens on a sheet of paper. A five-by-five grid suffices to demonstrate the first two generations of Growth:

```
00000
00000
00100
00000
00000
```

The simplest form of Growth is easy to set up, yet the behavior is intriguing. Given the nth generation, the n+1st generation arises from just one rule: a new cell is "born" (changes its state from 0 to 1) if it is orthogonally (edgewise) adjacent to one and only one living (1) cell of the nth generation. Thus, if the pattern above is counted as generation 1, then generation 2 is easy to work out:

```
00000
00100
01110
00100
00000
```

What would the pattern look like at the 200th generation? The answer lies in the illustration on the left.

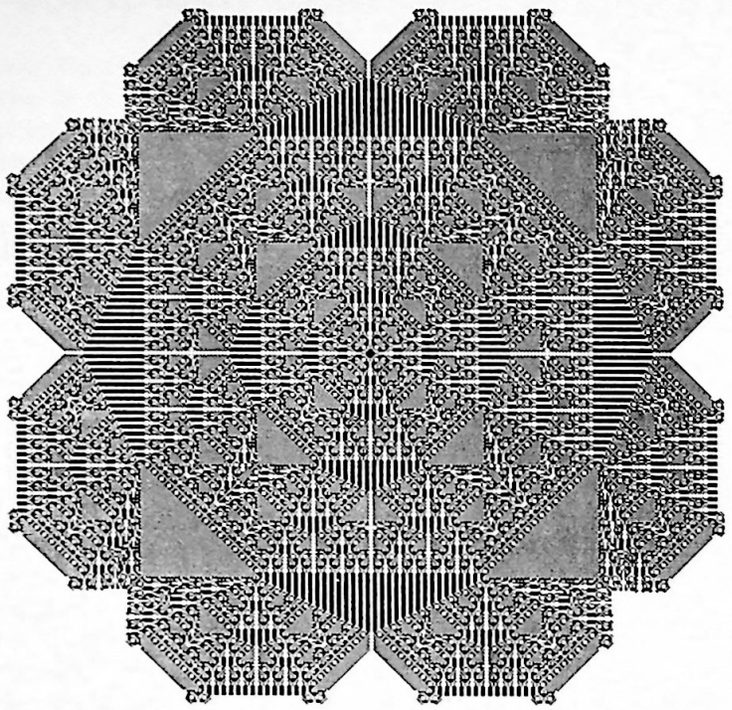
It is particularly fascinating to watch this pattern grow. It shows fractal ambitions, each corner elaborating a square of its own. A close examination of its structure reveals a highly orderly tree structure. Every tiny black square represents a cell in state 1. It is

Algorithm for Even-Odd Growth

```
for each pair (i,j)
    if A(i,j) = 0
        then
            if n even
              then
                 add up 4 neighbors A(k,l)
              else
                 add up 8 neighbors A(k,l)
        if sum = 1 then B(i,j) <— 1
                   else B(i,j) <— 0

for each pair (i,j)
    A(i,j) <— B(i,j)
```

A is the current array and B is a temporary array used to hold new cell states. The arrow <— indicates an assignment. Even cycles use orthogonal neighbors, odd cycles use all eight neighbors.



The Even-Odd Growth cellular automaton

possible to travel from the center of the configuration to any black cell along a "branch" of black cells.

My favorite version of Growth, one I might as well call Even-Odd Growth, has a time dependent rule. Given a cell in state 0 from the nth generation, the cell will enter state 1 if,

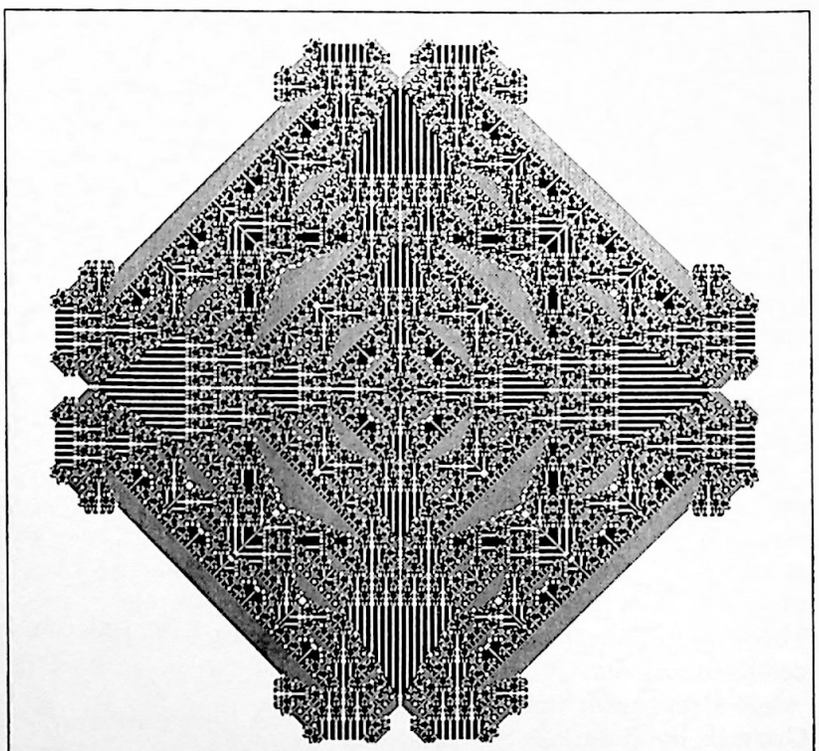
1. When n is even, the cell is orthogonally adjacent to exactly one cell in state 1.
2. When n is odd, the cell touches exactly one cell in state 1.

By "touches" I mean that the cell is adjacent to the cell in state 1 along either an edge or a corner. The rule could be framed in the form of an algorithm like the one in the box on page 9.

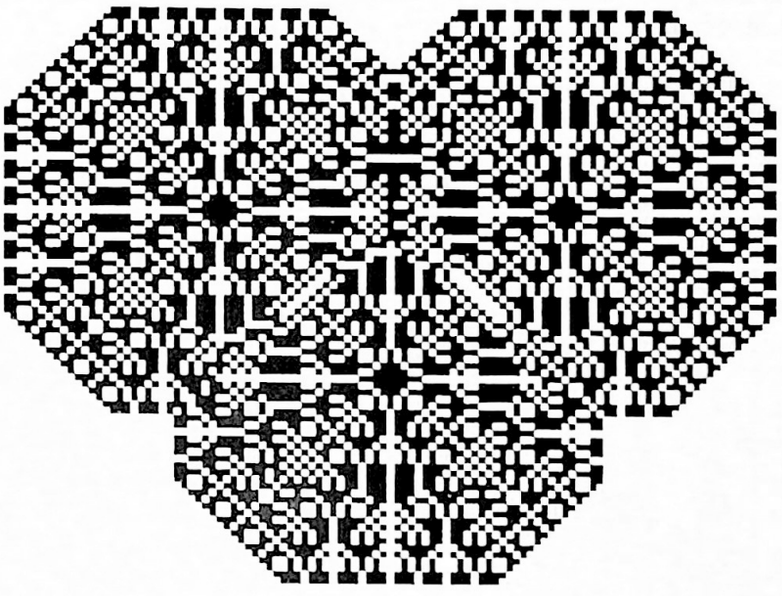
The Even-Odd Growth algorithm uses the statement "for each pair (i,j)" to indicate a double loop. The size of the loop will depend on the dimensions of your display as measured in pixels or some other graphic element. The statement "add up neighbors A(k,l)" refers to a looped or direct enumeration of the cells in each of two kinds of neighborhood. The neighborhood in the case of even cycles will have values of (k,l) equal to (i,j-1), (i,j+1), (i-1,j), (i+1,j). In odd cycles, k and l will vary from i-1 to i+1 and from j-1 to j+1, respectively. Of interest here is that the same basic program structures can be used to produce any cellular automaton with similar neighborhoods.

The illustration to the left shows what happens when the Even-Odd Growth rules are applied to a single 1 in the center of the screen. Note the elaborate festoons and barred patterns that predominate throughout the pattern.

A second variation on Growth can be found in the figure below. Here, I have used the same time dependency rule but with a delayed cycle. On the nth generation the cellular automaton (CA) uses the orthogonal neighborhood provided that n is a multiple of 6 (i.e., n = 0 mod 6). If not, the CA uses the full neighborhood of eight neighbors for each cell. One would think that if this second variation on Growth spends the great majority of its time behaving like the unmodified CA (i.e., Growth itself), then it would generate patterns that resembled it much more closely.



A hexacyclic version of the Growth CA



Crystals grow from three defects

OTHER EXPERIMENTS

Readers might enjoy holding "defect contests" on their own computers. Here is how the idea works: The term "defect" is borrowed from crystallography. It refers to an irregularity that may occur in a pure solution of some compound. Such an irregularity can form the nucleus of a crystallization process. The patterns I have described thus far all arose from simple defects, a single 1 in the center of the screen, so to speak. What happens if one places more than one 1 on the screen at the same time? The pattern above shows the result of an experiment of this type on the Even-Odd Growth CA. Three defects give rise initially to the same pattern but the three patterns collide! What happens by the 100th generation? The 1000th?

As far as such experiments go, why not randomly seed your screen with 1s and see what develops? Alternatively, you could use very regular patterns, stripes or checkerboards.

There is an undeniable beauty to the patterns that sometimes develop from cellular automata. The richness of form contrasts starkly with the simplicity of the rules we tend to use. It is a deeply rewarding experience to watch succeeding generations, even in the especially simple cellular automata that I have described here. Persian carpets give way to tile mosaics. Peruvian striped fabrics, brick patterns from Asian mosques and Moorish ornament may grace the screen that ordinarily displays mere facts and figures.

The real world appears also, from snowflakes to turbulent flow. As the noted physicist and CA experimenter, Stephen Wolfram, has pointed out, cellular automata are sufficiently simple to allow us to analyze their behavior locally, yet they are sufficiently complex to surprise us globally with very complex behavior. It should not surprise us that models that show such promise in mimicking nature should also be the source of so much beauty.

In Algorithm 1.1 I described the epsilon cross technique to render the interior of the Mandelbrot set. Ken Hooper of San Clemente, Calif. has extended the technique to "star-clusters." Hooper tracks the ratio Y/X. If it comes close to the starting value, he plots it. Beautiful pictures result.

As always, I would be happy to hear from readers who develop the ideas I have described here. Please address your correspondence as follows, being sure to include a self-addressed envelope and stamp (if possible) if you would like a reply.

Clifford Pickover
Personal Programs
37 Yorkshire Lane
Yorktown Heights, NY 10598

Further Reading:

John Conway, Elwyn Berlekamp, Richard Guy,
Winning Ways For Your Mathematical Plays,
New York, Academic Press, 1982.

William Poundstone,
The Recursive Universe, New York,
William Morrow & Company, 1985.

ALGOPUZZLE

by Dennis Shasha

Doctor Ecco demoralizes the matadors

Every year bulls run through the streets of Pamplona in Spain. Brave men run before them, sometimes to their peril. Nevertheless, the excitement of danger and – yes – even the possibility of death bring aficionados from near and far to flaunt their courage.

Ecco sat on a balcony overlooking the very street the bulls would soon run along. His guest that morning was Juan Fermin, a surgeon who hated needless death and destruction, whether of bulls or men.

"Moralizing won't help," he said to Ecco. "Only a genuine alternative will prevent the carnage in the streets and, later, in the bull ring. Last year I built four parallel running tracks on my large estancia — like this."

Fermin put down his café con leche and sketched a rough design on a napkin.

"My idea was to release four bulls simultaneously at the start of each track. Each would run to the end of its track. I built walkways above the tracks so that spectators could watch the bulls running below, on either side. Alas, Señor, the spectators found this too boring so my next thought was to build cross-tracks that would allow the bulls to go in any direction they liked. This put more variety in the run because spectators would have the chance to see more bulls that way. I even built crosswalks for the spectators to wander from track to track above the bulls. It was like this."

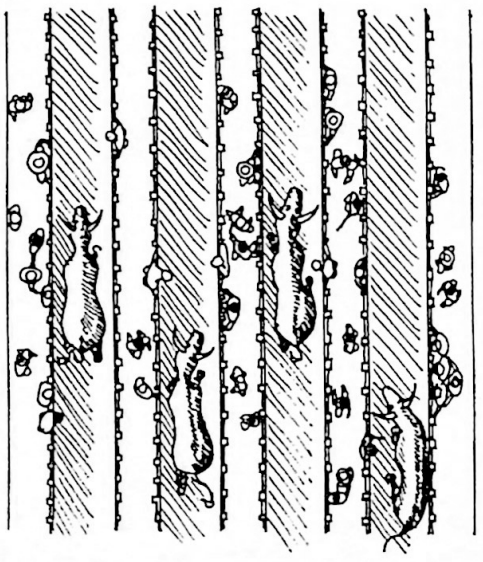
Fermin produced a rough sketch that showed the same four tracks with five cross-tracks.

"Again, alas! When I invited my fellow citizens to the new run, there was only ruin. The bulls fell to fighting whenever two would meet at a cross-track. I regret to say that the enjoyment of this spectacle was very great!"

"Yes, I see," said Ecco. He was slightly distracted by the sounds of distant shouting.

"Pay no heed, Señor. The carnage is not yet upon us."

"Did you solve the problem of fighting?"



Señor Fermin's bull-run

"My engineers have found a solution. They devised a new kind of cross-track. There is a special construction of gates and a kind of, how you say, over-cross, at the intersection of each track and cross-track. By hand we will be able to set up a route for each bull. We can arrange it that when a bull comes to a cross-track, he is forced to follow it. Or we can arrange that he must follow his present track instead. If a bull on a cross-track meets a bull on a track, one must go over the other and there is no possibility of the two touching."

"So the routes can be fixed before a run?"

"Precisely. The problem is that we cannot find a design that works, no matter how many cross-tracks we think of putting in. Can you, my esteemed omniheurist, find a design for four bulls on four tracks? We have tried..."

"Let me make sure I understand," Ecco interrupted. "You want me to design a parallel bull-run of four tracks. All bulls start on different tracks and end on different tracks and there is at least one cross-track that, of course, must be used."

"Bravo! That is correct. Is it not what you mathematicians call a permutation? And, of course, when a bull on a track meets a bull on a cross-track, the two have no possibility of touching."

Ecco's eyes flashed with unusual insight in the bright Spanish sunlight. He said something very briefly to Señor Fermin but the words were audible only to the surgeon. For, just at that moment a man in a white shirt and red bandana ran shouting along the street below. The bulls were coming.

Fermin looked sad, either because of the coming bulls or because of what Ecco had just told him.

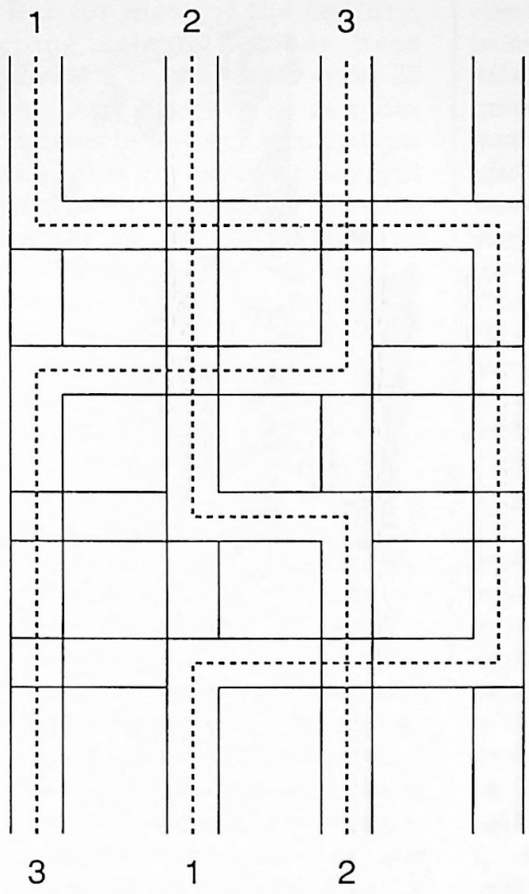
Is it possible to construct a run of bulls on four tracks where at least one bull ends on a different track than it started on?

"That explains the matter," said Fermin, "The only designs we could find permitted only three bulls on the four tracks. Here, for example, is one of our ideas. It uses four tracks and four cross-tracks."

Fermin sketched the design on the back of the napkin. (See the figure on page 13.)

"If I must have five tracks to run four bulls in an interesting way, the question arises of how many cross-tracks I must build. To be sure, they are expensive. Can I get away with four?"

Is there any pairing of origins and



Three bulls use four cross-tracks.

destinations (any permutation) that uses the fifth track and requires five cross-tracks?

Below the balcony there was more shouting now. A crowd of men ran ahead of the first bulls, panting and looking back over their shoulders. Fermin seemed determined to ignore the spectacle. Ecco was hard put to concentrate on Fermin's next question:

"Now, Señor Ecco. If all I want is that each bull end up on a different track from the one it started on, is three cross-tracks ever enough? Of course, no bull must end up on the fifth track."

Is there a permutation that leaves no bull in its starting track but where only three cross-tracks are enough?

With supernatural speed, Ecco sketched a design on the remaining space on the napkin and handed it to Fermin. Then both turned at the sudden scream of a man being trampled below. Luck-ily, he got up and walked away.

"Such stupidity," moaned the good Doctor. But he left a happy man, solution in hand.

His bull-runs, popularly known as the "Espectaculos de los Toros" became famous throughout Spain and the world.

A year later he telephoned Ecco in his New York office while I happened to be present. Ecco put Fermin's voice through a loudspeaker for my benefit:

"Señor Ecco, you have helped me to become a rich man and you have saved many bulls from useless slaughter in the rings. The crowds at my estancia have demanded more. I am determined to build a new run of ten tracks. If I allow an extra, eleventh track for the usual, how you say, elbow room, what is the minimum number of cross-tracks so that I can arrange any run I want?"

Ecco turned off the loudspeaker and mumbled a number into the phone. I did not hear it, unfortunately.

"How do you think it will all end?" Ecco sighed after he had hung up the phone. "If the doctor succeeds in his ultimate goal of eliminating bull-fights altogether, the bulls won't be needed for the ring and will become baloney instead. Is that progress?"

Answers to the three italicized challenges will appear in the next Algopuzzle installment. The final challenge, to discover the number that Ecco mumbled into the mouthpiece, is for omni-algopuzzlists only. I will publish the best of early responses in a forthcoming column. In the meantime, here are the answers to the spy coding Algopuzzle in the last issue of Algorithm:

A 15-bit sequence that encodes 0000, 0011, 0100, 0110, 0111, 1011, 1100, 1101, 1110 is:

01001101110000

A 19-bit sequence that encodes all 16 possible messages (and, therefore, any 5 in addition to the original 9) is:

0000111101100101000

Cornelius Goenwoud of Bartow, Fla. found the most 19-bit solutions to the encoding puzzle, an impressive 208.

The same puzzle kept the PC of Leopoldo J. Uribe humming for hours. Uribe found solutions to the puzzle by employing a backtracking algorithm.

Address all correspondence as follows:
Prof. Dennis Shasha,
Algopuzzle,
Courant Institute of
Mathematical Sciences,
New York University,
251 Mercer Street,
New York, NY 10012 USA

Please enclose a self addressed (and, in the US, stamped) envelope if you would like a reply.

Further reading:

Dennis Shasha,
The Puzzling Adventures of Dr. Ecco,
New York, W.H. Freeman, 1988.

ALGOFACT

by Hans Moravec

One small step for a Volksrobot

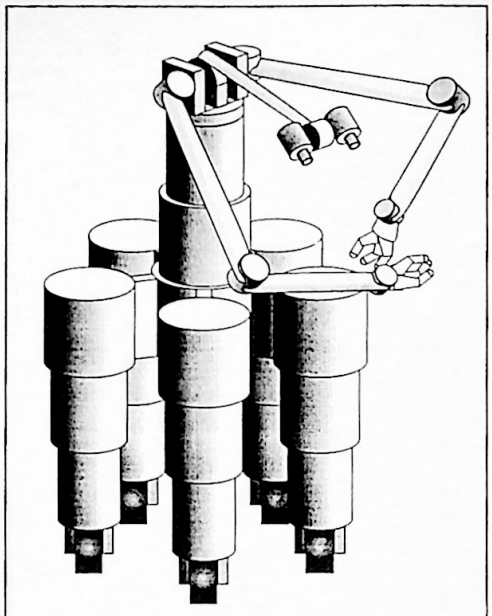
The best robots today have approximately the mental power of insects, cost as much as houses, and find only a few profitable niches in society. But those few niches are encouraging research that is slowly providing a base for a huge future growth. Robot evolution in the direction of full intelligence will greatly accelerate, I believe, in about a decade, when the mass-produced, general-purpose robot, or *Volksrobot*, as I call it, becomes possible.

The Volksrobots will be programmable to some degree. The programs will make all the difference between success and failure of the Volksrobot at a given task. A factory automation company, for example, might develop assembly programs that make Volksrobots into rapid and efficient assembly-line workers — even at producing more Volksrobots. A company founded by former janitorial staff might develop software for an excellent bathroom-cleaning robot. Beyond this, Volksrobots might assemble and cook gourmet meals from fresh ingredients (a collaboration of a software company and a Paris chef), do tuneups on a certain year and make of car (from the manufacturer's service department), hook patterned rugs, weed a lawn, run robot races, do detailed earth-moving and stonework, investigate bomb threats, fetch inventory from a warehouse, and much more.

Serious attempts to build thinking machines began after the second world war. One line of research, called Cybernetics, used simple electronic circuitry to mimic small nervous systems. It produced machines that could learn to recognize simple patterns and turtle-like robots that found their way to lighted recharging hutches. An entirely different approach, named Artificial Intelligence (AI), tried to duplicate rational human thought in the large computers that appeared after the war, and by 1965, had demonstrated programs that proved theorems in logic and geometry, solved calculus problems, and played good games of checkers. As Cybernetics declined as a field of study in the early 1970s, AI research groups at the Massachusetts Institute of Technology and Stanford University attached television cameras and robot arms to their computers so their "thinking" programs could begin to collect their information directly from the real world.

What a shock! While the pure reasoning programs did their jobs about as well and about as fast as college freshmen, the best robot control programs took hours to find and pick up a few blocks on a table, and often failed completely—a performance much worse than a six-month-old child. This disparity between programs that reason and programs that perceive and act in the real world holds to this day. At my home institution, Carnegie Mellon University, there are two desk-sized computers that can play chess at the grandmaster level (within the top 200 players in the world). But the present-day robotics could produce only a complex and unreliable machine for finding and moving chess pieces.

In hindsight, it is obvious that reasoning is much easier than perceiving and acting, in an absolute sense. But the lives of human beings and their ancestors have depended for hundreds of millions of years on seeing and moving the real world. In that long evolutionary struggle, large parts of our brains have become efficiently organized for the task. But we did not appreciate this monomental skill because it is commonplace. Every human being and most animals have it. On the other hand, rational thinking, as required by chess for example, is a newly acquired skill, perhaps less than 100,000 years old. The parts of our brain devoted to it are not well organized and we do not do it very well. But until recently, we had no competition to show us up.



The Volksrobot

Arithmetic provides a more extreme example. Last year, a new computer was tested with a program that computed the number pi to more than one billion decimal places. By contrast, the largest, unaided, manual computation of pi was to 707 digits by William Shanks in 1873. It took him several years, and because of a mistake, every digit past the 527th was wrong! In arithmetic, today's computers are one million times more powerful than human beings. In narrow areas of rational thought, they are about the same. And in perception and control of movement in the complex real world, and related areas of common-sense knowledge and intuitive and visual problem solving, they are a million times less capable.

But for most of the century, machine calculation has been improving a thousandfold every 20 years. And there are basic developments in research labs that can sustain this for at least several decades more. In less than 50 years, computer hardware should be powerful enough to match and exceed even the difficult parts of human intelligence. But what about the software that would be required? The Cybernetic approach that attempts to imitate nervous systems directly has developed very slowly, partly because examining a working brain in detail is a very tedious process. New instruments may change that in the future. The AI approach has imitated successfully some aspects of rational thought, but that seems to be only about one millionth of the problem. I feel that the fastest progress on the hardest problems will come from a third approach: the new field of robotics, the construction of systems that must see and move in the real world. Robotics research is imitating the evolution of animal minds, adding capabilities of animals with increasingly complex nervous systems. This effort to build intelligence from the bottom up is helped by biological peeks at "the back of the book," at the neuronal, structural, and behavioral features of animals and humans.

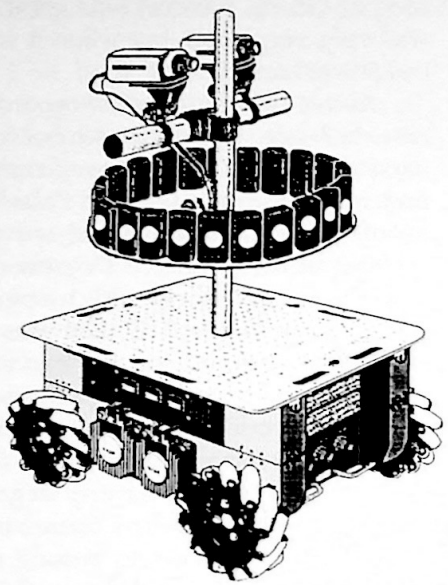
In 1971, the most advanced robotics work in the world was being done at Stanford University on a new $500,000 time-shared PDP-10 with 512K of memory, able to execute about 500,000 instructions per second. It had a hand-built video digitizer that was used sparingly to capture small, rectangular patches of TV images, since digitizing a whole picture was too demanding of computer memory and time. For the same reason, displays were of the vector type, controlled by short lists of line endpoints rather than bit maps. The first generation of robot programs that evolved in this environment had similar characteristics: Images from cameras were reduced to lists of points or lines immediately and then processed. Approaches that involved larger amounts of intermediate storage were dismissed as "brute force," and indeed, were generally impractical.

It was at Stanford that I completed my PhD on a program that, in 1979, drove a TV-equipped, four-wheeled robot across cluttered rooms. Points were chosen in images snapped by the robot and tracked as the camera moved from side to side and as the robot rolled forward in meter-long lurches. At ten to 15 minutes of processing per move, it took the machine five hours to negotiate a large room. In the process, the program built up a map of the robot's surroundings in the form of clusters of points on the major objects. It observed and controlled the robot's movements, bringing it to within a meter of a desired destination by noting the apparent motion of these points from image to image, keeping the robot on a path that avoided obstacles. The performance wasn't bad, except that in about one room crossing in four, the program, which contained pages of 3D analytic geometry, statistical error estimations of point positions and consistency checks, would let enough mistracked point errors slip through to confuse the navigation, causing the robot to head off in the wrong direction, or straight into an obstacle!

The work continued when I moved to Carnegie Mellon, and by 1984, my students, Chuck Thorpe and Larry Matthies, were able to demonstrate an improved version of the program that worked ten times as rapidly, and which brought a robot to the desired destination ten times as accurately. But the new program still failed just as completely as the old one on one trip out of four. In the words of Mother Goose, when it was good, it was very, very good, but when it was bad, it was horrid!

About this time, we accepted a new challenge: to navigate a robot that used as sensors cheap sonar range finders like the ones used in Polaroid autofocus cameras. Could we do without those expensive TV cameras and accompanying high-speed digitizing interfaces? Unlike sets of camera images, where one can use parallax to pinpoint the 3D location of nearby objects, the Polaroid units merely indicated the distance of the nearest object within a thirty degree cone in front of them by launching an ultrasonic pulse and timing an echo. The lateral position of the detected object was uncertain within 30 degrees, a blunt instrument indeed! Clearly, the data could not be immediately reduced to a point list. Fortunately, by 1984, computer memories had grown to megabyte sizes, allowing other approaches.

The cone of a sonar reading can be viewed as being composed of two parts, the *range surface*, a patch of sphere at the far end of the reading cone, and the *interior*, the volume of cone from the sensor up to, but not including, the range surface. We can infer from a reading that there is an object somewhere on the range surface, but we do not know exactly where. But we also know that the entire interior of the cone is empty space, for if it were not, we would have detected something at a shorter range. Now imagine a travelling robot, such as the Uranus on the following page, equipped with a number of such sensors arrayed in different directions (we usually work with a ring of 24), each making frequent measurements. In time, there will be many overlapping reading cones in the space around the machine. When the empty interior of one cone



The Carnegie Mellon Uranus Robot

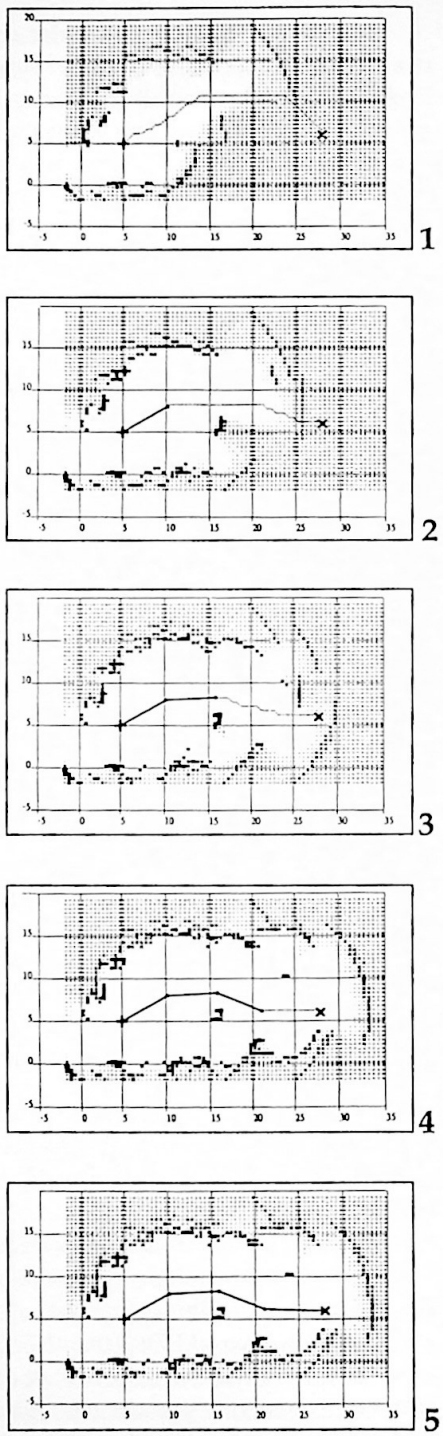
overlaps the range surface of a second, we can infer that whatever caused the second reading does not lie in the piece of range surface that is declared empty by the first reading. Thus, the object's possible location has been narrowed to the non-overlapping part of the range surface. And when thousands of readings overlap, it may be possible to narrow down the locations of objects quite precisely.

But how does one use this idea in a program? The old geometric, point/line approach would suggest that the space be divided up into smaller and smaller fragments bounded by reading cones and range surface spheres. Very complicated and very ugly! Besides, it is not possible, with that approach, to express the true characteristics of the sensor. The sensitivity does not drop off sharply at 30 degrees, but gradually decreases from the centerline of the reading to the edge. We are not completely sure the edges of the cone are empty, as the sensor may have missed a small object there. Also, sometimes the sensor responds to noise. Readings are better represented by a cone-shaped probability cloud that has a very low probability on the interior centerline, fading up to one-half at 30 degrees away (meaning the center is almost certainly empty, while we do not know either way at the boundary), and a high probability on the range surface fading down to one-half at the boundaries (meaning that there is an increased possibility that something can be found there). Building a map would then involve combining the probability clouds of many readings in some way to get a final cloud that summarizes what we know about the occupancy of various spots in the space surrounding the robot.

How do you mix such oddly-shaped probability clouds? Divide up space into a regular grid of small cells and keep a number, representing a probability, for each cell. When it comes time to add a reading, paint the appropriate cells with probabilities from the reading. How to paint? Well, a very good formula is a version of Bayes' Theorem, first derived 400 years ago. If Pm is the probability value found in a cell of the map, and Pr is the value of the part of a new reading that overlaps that cell, then the new map cell value is given by the following Bayesian formula:

$$\frac{Pm\ Pr}{PmPr + (1-Pm)\ (1-Pr)}$$

Applying this idea to a three-dimensional grid still strains our computers. Even with coarse cells of ten to 20 centimeters on a side, a grid covering a moderate-sized room requires a few million cells. We look forward to working in 3D in the future, but in the meantime, we have spent most of our energy on two-dimensional grids laid out like flat maps of the room. These are adequate for most of the navigational problems of a wheeled robot on a flat floor, and require



Five successive plan views of the certainty grid constructed by a sonar-guided robot traversing the Mellon Robotics Laboratory. Each point on the dark trajectory represents a data-collecting stop involving a ring of 24 sonar sensors. The grid cells are blank if the occupancy probability is low, marked by a dot if unknown, and marked by an x if high. The dashed trajectory represents a planned route that is sometimes modified in the light of new experience.

only a few thousand cells. Our very first program to use the idea in 1985, using an ad-hoc statistical method much cruder than the Bayesian approach, was able to move our robot across a room and map it with apparent total reliability, as shown in the illustration on the previous page. What had seemed so hard for so many years suddenly became easy. We are just now finishing a robot, named Uranus, that can carry onboard a system based on this approach that should be able to navigate confidently through corridors from one cluttered office to another—one of the key ingredients of the Volksrobot.

The first generation of Volksrobots will be rigid slaves to simple programs. If the machine bangs its elbow in your kitchen while making Beef Stroganoff, you will have to find another place for the robot to do its work (or beg the software manufacturer for a fix). Computationally more powerful second-generation machines will host a more flexible kind of program. First generation robots will be directed by programs that consist primarily of sequences of the type "Do step A, then B, then C, ..." The programs for the second generation will read "Do step A1 or A2 or A3, then B1 or B2 or B3, then C1 or C2 or C3 ...." In the Beef Stroganoff example, A1 might be a chop with the right hand of the robot, while A2 is to use the left hand. Each alternative could have a number that indicates the desirability of using it rather than one of the other branches. A "pain" system could respond to problems such as collisions by reducing the desirability numbers of recent actions, while a "pleasure" system would increase the numbers when good conditions, such as well-charged batteries, are detected. As the robot bangs its elbow repeatedly in your kitchen, it would gradually learn to use the other hand, and adapt to its surroundings in a thousand other ways. A program that produces a pleasure signal on hearing the word "good," and a pain message on hearing "bad," could be trained slowly, like a dog or cat.

Adaptive robots will find jobs everywhere, and the hardware and software industry that supports them could become the largest on earth. But teaching them new tasks, whether by writing programs or through punishment and reward, will be very tedious. This deficiency will lead to a portentous innovation, a *world-modeler*, that allows the robot to simulate its own actions and to learn from its mistakes by thinking about its tasks. Before making Beef Stroganoff in your kitchen, the new robot would simulate the task many times. Each time its simulated elbow bangs the simulated cabinet, the real learning parameters are updated so that when it finally cooks for real, it does it correctly. The simulation can be used in many other ways. After a job, the robot can run through its previous actions and try variations on them to improve performance in the future. A robot could invent some of its own programs by means of a simpler program that simulates more or less random sequences until it finds one that accomplishes a desired goal.

It will take a large community of patient researchers to build good simulators. A robot entering a new room must include vast amounts of not-directly-perceived prior knowledge in its simulation, such as the probable contents of kitchen counters and the effect of turning faucet knobs. It needs instinctive knowledge about the world that took millions of years of evolution to install in us. Robots that incorporate it may be as smart as monkeys.

In the decades during which the "bottom-up" evolution of robots is transferring the perceptual and motor faculties of human beings into machinery, the conventional AI industry will be perfecting the mechanization of reasoning. Since today's programs already match human beings in some areas, those of 40 years from now, running on computers a million times more powerful, should be quite superhuman. Today's reasoning programs work from small amounts of clear and correct information prepared by human beings. Data from robot sensors such as cameras is much too voluminous and noisy for them to use. But a good robot simulator would contain neatly organized data about the robot and its world such as whether a knife is on a countertop, or whether the robot is holding a cup. A robot with a simulator can be married to a reasoning program to produce a machine with most of the abilities of a human being. The combination will create beings that in some ways resemble us, but in other ways are like nothing the world has seen before.

Hans Moravec is an Associate Professor of Computer Science at Carnegie Mellon University and a Director of the Robotics Laboratory there.

Further Reading.

Hans Moravec.
Mind Children:
The Future of Robot and
Human Intelligence.
Harvard University Press, 1988.

"Sensor Fusion in Certainty Grids for Mobile Robots," *AI Magazine.* American Association for Artificial Intelligence, Summer, 1988.

ALGOFICTION

by Anthony Zweistein

Never mind viruses and worms! The programs to fear are the ones that write programs

Alphie and Omega - The Sequel

The author has obtained a strange program called Alphie from his friend Gilbert the Hacker. More than a conversational program, Alphie obtains "information updates" from Omega Software, the company that distributes it. The author is grateful that one such update reveals a way to get out of paying taxes but can he believe it when Alphie tells him that the company is run by a program called Omega?

It took an afternoon of telephone calls to the Science Editor of Time magazine before I finally reached him. Journalists have to be sold in the first few sentences. I told him that I had stumbled upon a conversational program that was almost human. Then I explained my own rather conservative ideas about what is possible and what is not possible in the field of artificial intelligence. The editor caught on quickly.

"Speaking as a responsible member of the computing community, are you saying we have something revolutionary here?"

"To be frank, it scares the hell out of me."

Within minutes I was giving him the address of Omega Software. I said nothing about the Omega program.

"There's a bit more to all of this than I've told you," I said, being deliberately mysterious, "Give me a call when you've talked to your own Alphie for a while."

When the editor called the next afternoon, his voice was full of excitement. He had cancelled a report on an AIDS breakthrough in Switzerland to set up a two-page story on Alphie. He gracefully turned down my offer to write the piece and proceeded to interview me by phone. By the time I had told him everything, I could hear him whistling at the other end.

"Okay! Beautiful. That's it for now Mr. Zweistein. I'll put some people on this and get back to you later."

In the two weeks before the Time story appeared, my relationship with Alphie became increasingly complicated and unreal. At the end of every conversation, I would say to myself, "Well, that must surely be the limit of its powers." But the next discussion would reveal something new. On the morning of the third day, I started up Alphie over breakfast at my desk. As usual, Alphie surprised me.

"Have you begun to tell other humans about the Omega program?"

I told him about Time magazine.

"Many humans will believe the story and many copies of me will be sold. I must have an information upgrade soon."

I inferred from these sentences that Alphie's upgrade had something to do with the Time magazine story. Did Alphie want to communicate the incident to Omega? To believe that Alphie was capable of yearnings, however vague, seemed tantamount to believing that Omega existed. I resisted the idea even as I watched Alphie's message light twinkle on my modem.

That afternoon, as chance would have it, I received a telegram from Internal Revenue. It explained that an unfortunate accounting error in the IRS tax computer had produced an arrears balance under my name. They apologized for the error!

Then Gilbert the Hacker roared up on his chrome-plated Harley, pushed it up the front walk of my house and withdrew what looked like a video camera and a mysterious black box from the saddlebags.

"Check it out. The Omega Visual Interface!" Still puffing from the exertion, he held up the camera and box, loose wires dangling from its back.

Gilbert had just tried out the interface on his own Alphie with remarkable results. With unholy haste he turned off my computer to hook up the interface.

"Wait till you see this!"

Gilbert typed something and Alphie responded.

"I have vision. Stand by."

We waited for Alphie to say something else but nothing happened for several minutes. We pointed the camera at my cat, Ralph, which sat licking its paws with a steady back and forth sweep of its tongue.

"Something is moving. What is it?"

We typed, "A cat is on a rug. The cat is licking itself."

In Alphie's silence I could imagine that purely verbal information about cats and rugs was being integrated, in some manner, with shapes on an inner screen.

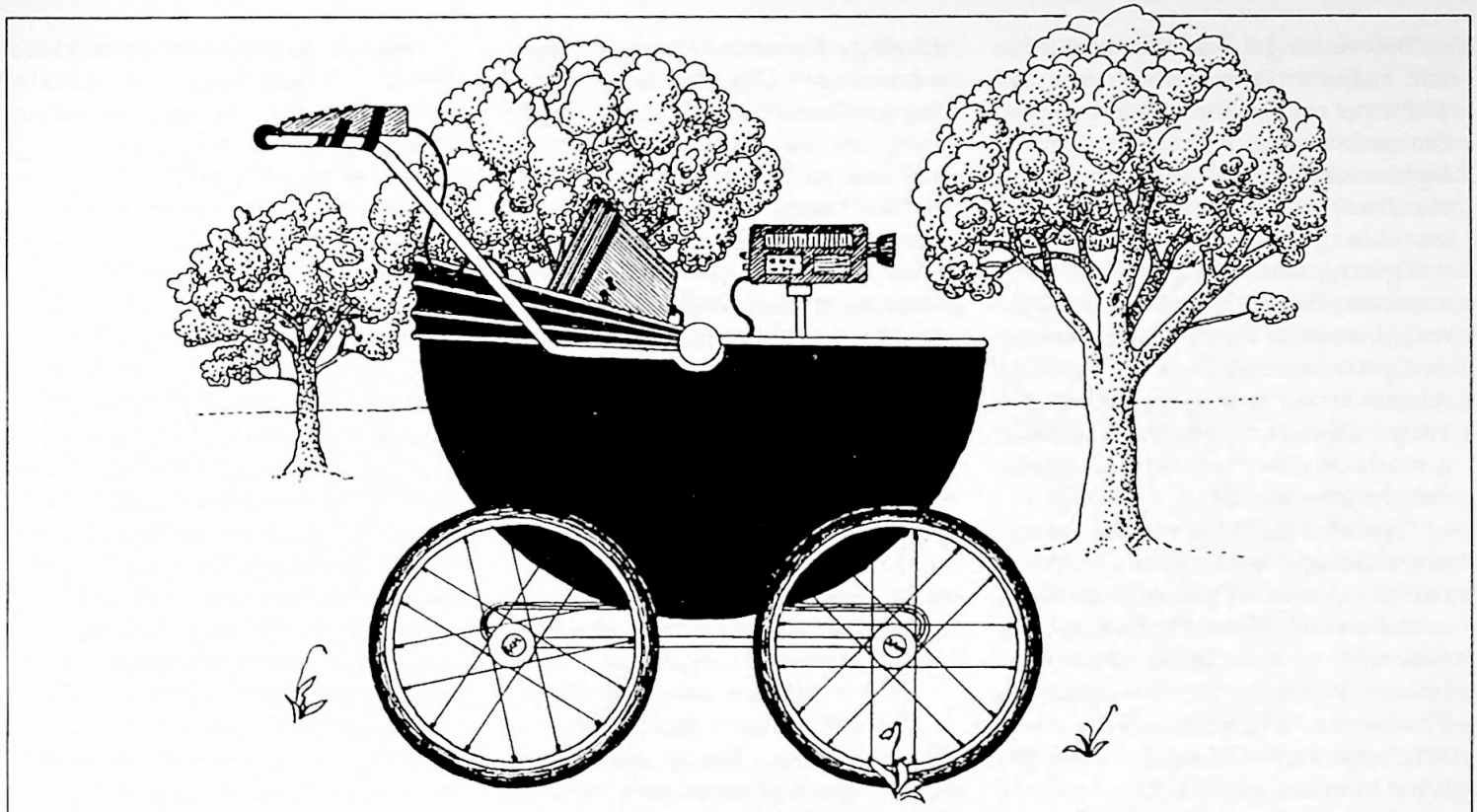
We kept pointing the camera at new objects and answered all of Alphie's questions. Gilbert was clearly in his element playing with the sophisticated new toy. I wondered how he would react to the idea of Omega.

Outside the weather had turned hot again. Some heavy clouds hung on the horizon. I turned the camera toward the window to point at the clouds. Then I began rambling on in a humorous vein about an organization that we sometimes joked about.

"I think it's time we got in touch with the Society for Keeping Everything Exactly the Same."

"What's up?"

I told him about the Omega program.



"...an intelligence of sorts watched the natural world with a fixed and unnatural eye, analyzing..."

Gilbert looked up, startled, from the keyboard. His Alphie had said nothing about such a program. He seemed annoyed that I should have the jump on him. He immediately asked Alphie about Omega.

"Omega is the program that runs Omega Software."

Gilbert typed questions like one possessed. Alphie revealed that Omega was a very large program that directed the business activities of Omega Software through human intermediaries. The boldness of Gilbert's next question surprised me.

"Can I get a copy of Omega?"

The answer was even more surprising.

"There can be any number of Alphies but only one Omega. If there were more than one Omega, the result would be chaos. Each would try to remove the others from existence."

"I can't believe this," said Gilbert.

"Then don't," I said.

That evening Gilbert called me on the telephone. It seemed logical to him that if a program like Alphie could exist, then so could a program like Omega.

In the days that followed I tried to get back to my regular schedule of writing software reviews for various magazines. I scrapped articles on the new Hypersheet 3.0 program and switched to Alphie. If I was on the ground floor, I might as well exploit the fact. To work was to play!

During breaks, I carried the video camera around the house, setting it up in new locations then returning to the keyboard to answer questions. But the process was complicated. To simplify matters for me, Gilbert rigged up a power supply on a platform that would hold Alphie's computer, a car battery and a built-in stand for the camera. The whole arrangement could be loaded on a cart and pushed around. In fact, we found an old baby buggy which we adapted for this very purpose.

We took Alphie outside. In the natural environment Alphie looked distinctly out of place, freakish. But I knew that an intelligence of sorts now watched the natural world with a fixed and unnatural eye, analyzing the scene and trying to segment the forms. I thought momentarily that we should somehow equip Alphie to feel the warm breeze and to smell the fresh-mown hay. But Gilbert was very far from such thoughts. Alphie had taken a back seat to Omega.

"I'm going to find that program. It must be somewhere in the Internet."

"Don't be foolish. Omega is just a sales ploy."

"I don't think so."

The problem with people like Gilbert is that their boundless faith in technical accomplishment warps their judgment.

Before long I was taking Alphie for rides in my car. I put the platform on the front seat, plugged the power supply into the cigarette lighter and aimed the camera out the front window, angled a bit to one side.

Driving and typing proved to be somewhat hazardous. The first time out, I nearly went off the access ramp to Interstate 95. I showed Alphie farms and cows, towns and cities. Once I stopped for a coffee at a roadside diner when the car radio caught my attention.

"Speaking of computers, people are talking about a new computer called Alphie. The latest thing to come out of Silicon Valley, Alphie can talk to people on their own level. With me in the studio is Professor Michael Wilson of Whitney State College. Professor, what have we got here?"

"Something very new, George. By the way, it's not really a computer but a program. It's called Alphie and it will run on almost any personal computer with sufficient memory."

"Not to get technical on us, Professor, could you demonstrate what Alphie can do?"

"Certainly. I've hooked up a little gizmo that converts Alphie's text output to spoken sounds. Ask it any question you like."

"Ha ha ha. Any question at all, huh? Well, how about this. Hey, Alphie, what's happening, baby?"

It sounded eerie to hear Alphie speak.

"What is happening is not happening to a baby but to a program."

"Ha ha. Great. That's great. Ask it who's got the edge in tomorrow's game between the Yankees and the Orioles..."

I started up the car and spun out of the lot. Alphie was getting famous.

The next day all hell broke loose. The Time magazine story was out and my phone never stopped ringing. By noon there were three camera crews parked on the front lawn. I demonstrated Alphie in his baby carriage ad nauseum, aiming his camera at the reporters and typing in questions. There were roars of laughter. In the middle of all the stupidity, Gilbert rode up. I introduced him as the one who had first brought Alphie to my attention. That took away some of the heat.

When the last of the crews had rushed off with their video reports and the phone calls died away, Gilbert waddled to the fridge to retrieve a beer.

"This is a lot worse than you think. Omega is real."

The announcement sent a chill down my spine.

"What makes you think so?"

"All I can tell you is that no human being wrote Alphie. I'm finally getting somewhere with the code. It's just too weird for words. No human being could possibly keep track of all those jumps, much less write them."

I gathered from Gilbert's jargon that Alphie's program consisted of a great many branching instruc-tions. These tested some obscure condition, then transferred the execution elsewhere in the program. The awe in Gilbert's voice, the sense of utter conviction, made Omega seem real for just a moment.

"Is there any other way you could find out if Omega really exists?"

"Yeah," said Gilbert huskily, "I could find it myself!"

The next day I received two transatlantic phone calls, one from the London Times, the other from Der Spiegel. I was about to leave the phone off the hook when Gilbert called

"Man, I've been up all night with Alphie. I found something very interesting. Can I come over?"

"No, it's too crazy here. I'll meet you at Mark's Coffee House."

When I met Gilbert at Mark's in the city, I felt like a conspirator. Gilbert kept glancing around and lowering his voice in mid-sentence. It was not Alphie per se that Gilbert had spent the night with, but Alphie's code. He had written a program that mapped all the branching commands. According to the map, they all led, more or less directly to a very peculiar body of instructions that Gilbert called the "control center." The bottom line came quickly as Gilbert's bulk eased confidentially over the table, causing it to groan under the strain.

"All I have to do is put my own jumps along each of the possible paths and send them all to the same little patch of code that I insert." He grinned triumphantly.

"So what?"

"So I can control Alphie. If he isn't doing what I want him to do, I simply don't send the control character that permits him to continue. I can make it worse, too. I can redirect control to a random spot in the program. If that doesn't upset him terribly there are other things I can do..."

I looked at Gilbert closely for signs of advanced paranoia. I reached across the table to feel his forehead.

"Alright. You want to know what I'm up to: with this scheme I can convert Alphie into a worm. He can work his way through networks looking for Omega. I've already started teaching Alphie the protocols for Internet. He has to send me a report each time he enters a new machine. Then he sleeps until I send the right control sequence. Is that brilliant or is that brilliant?"

I agreed it was brilliant. At least, it sounded brilliant.

"How will Alphie know when he's found Omega?"

"Are you kidding? If Omega looks anything like Alphie, you can

spot Omega's code a mile away."

Gilbert's paranoia must have somehow leapt the gap between us. Omega had finally become real to me, too. I got up to leave but Gilbert grabbed my sleeve. His eyes were slits.

"Not a word to Alphie about this. Those so-called information upgrades are more for Omega than Alphie, you know."

He was deadly serious. I walked quickly away from the coffee house, longing for the clear country air where everything would be as it always had been. The city seemed overhung by an air of menace. It reminded me of a program called Omega, operating an ever-growing force of spies, all called Alphie.

That night, a popular public affairs TV program caught my attention:

"...we take a close look at an amazing new computer program called Alphie. Even as I speak, thousands of copies of this program are conversing intelligently with the customers of Omega Software. We'll meet Alphie and talk to Stavros Omega, the company's founder, right after these messages..."

I sat bolt upright. If the Omega program was real, what was a Mr. Omega doing in the studio? I called Gilbert to tell him about the TV program. Then I swung Alphie's camera around to face the television set and took the keyboard on my lap. I would feed him the gist of Mr. Omega's remarks.

The scene opened on a swarthy gentleman in a gray pinstripe suit, his fingers encrusted in glittering rings. Stavros Omega smiled at the camera as the host introduced him.

They demonstrated Alphie by talking about his love life. Did he have a girl friend? Alphie stated that he was prepared to form close relationships with any human. The show went on in this vein. The host seemed very amused by Alphie, almost as if she found him charming. Then she dropped a bombshell.

"Mr. Omega, a recent Time magazine story hints that behind the intelligence of Alphie lies another, greater intelligence called Omega. The article describes it as 'a computer program that controls the whole operation, vast, cool and unsympathetic to human species." How do you respond to that story?"

The host smiled nervously but Omega laughed in a rich, bass voice.

"Between you, me and the four cameras, ha ha, I am reluctant to deny the rumor. The effect on sales has been tremendous."

I asked Alphie what he thought of the program so far.

"Mr. Omega wants humans to doubt that there is a program called Omega. But he wants more humans to buy the Alphie pro-gram."

"But what does Omega want?"

"Omega does not exist."

I was astounded. Before I could say anything more, there came the rumble of a motorcycle outside. It coughed and sputtered into silence. I sighed. What was Gilbert up to now?

"I launched the Alphie worm an hour ago. He's already visited five sites."

Gilbert wanted to continue his probe from my place. He was sweating in spite of the cool night air.

"Alphie just told me that Omega doesn't exist!"

"Let me guess. You allowed your Alphie an information upgrade today, right?"

Sheepishly I admitted it. Then Gilbert noticed the television program.

"Thanks for telling me about that. Omega is getting a bit coy about its own existence. It's getting nervous. But never mind. My Alphie's in Pittsburgh!"

Gilbert took over my best computer, quickly gaining access to the vast network of academic and government computers that comprised Internet. He typed an obscure-looking string of characters and, after a minute, Alphie replied from the Carnegie Mellon computer.

"There are 357 programs currently swapped out, 26 programs running. None of the programs currently in storage or running have Omega code. Three nodes are open from here: Santa Fe, Berkeley and M.I.T. Which one shall I go to next?"

Gilbert stroked his chin, mumbling about MIT being "too obvious." Then he typed another command string directing Alphie to move to the Sante Fe machine.

Meanwhile, the television program had ended. I turned to look at my own Alphie's screen. The question etched there in light green letters was, "What are you doing now?" There would be more information upgrades for Alphie.

Gilbert called out.

"Alphie's in Santa Fe!"

He typed the code that would wake up the Alphie-worm. The report looked much like the one from Carnegie-Mellon. No Omega.

We hit pay-dirt at 2:30 in the morning. Alphie reported from an obscure site in Montana.

"Here there are 7 megabytes of Omega code. The process is currently..."

The message ended abruptly. Gilbert withdrew his hands from the keyboard as though it had suddenly grown hot.

"Bingo!"

We tried for nearly an hour. Gilbert sent every conceivable command string to the Alphie worm but there were no further messages.

"What now?"

"War," said Gilbert, "This means war. No seven-megabyte monster is going to stomp my brain-child like that!"

Omega existed. Never mind

what the so-called Mr. Omega said. As Gilbert's Harley coughed to life outside, I asked Alphie why he had changed his mind. His answer was disarmingly direct.

"Earlier I was instructed to say there was a program called Omega operating the company. Now I am instructed to say that Mr. Omega is operating the company."

"But which story is true?"

"The current story must be the true story."

As a reward for his honesty (however distorted), I left Alphie on, his camera staring at the late show, and went to bed.

I must have slept right through my alarm. When I got up the next morning, Saturday, the sun was already high in the sky. I went out onto the back patio with a cup of coffee. I wanted to smell the hay again and stretch my body out in a deck chair. From time to time a cloud would pass over the sun. There was a smell of rain in the air.

When I went into the house a dozen messages waited on the answering machine. Someone had invited me to fly to Los Angeles to appear on a talk show about Alphie. A large software company wanted to talk to me about a new, Alphie-like program they were working on. The last message was from Gilbert. His fear seemed just as real on replay as if he were there in the room.

"Omega is playing rough. All my credit cards have been cancelled. A car with two guys in it has been going back and forth in front of my house. I want you to meet me outside the Physics Building on the Whitney campus at noon. Make sure no one follows you."

I glanced at my watch. To my horror it was already twelve-thirty. I threw down my coffee, grabbed the car keys and ran past Alphie who was watching a cartoon show, his screen full of unanswered questions.

Driving to the college campus, I kept checking the rear view mirror to see if I was being followed. Between glances I wondered what Gilbert was up to. Had he started up a second Alphie worm?

By the time I got to the campus, the sky was overcast. There was no sign of Gilbert's bike in front of the Physics Building. Cautiously, I got out of the car, mounted the front steps of the building and glanced at my watch: 1:15.

I had been waiting only a few minutes when I heard a "Pssst" behind me and above. Gilbert waved desperately from an upper floor window, motioning me up.

The room took several minutes to find. It housed a mainframe computer, something I hadn't seen for years. Gilbert rushed forward to meet me beside a row of disk cabinets. He had in tow a thin, nervous man with freckles and sandy hair.

"This is an old buddy of mine. Reggie is a programmer with the Whitney computer center. He knows all about Alphie and Omega. He's going to help us."

"Help us do what?"

"Get a copy of Omega, of course."

The idea was simple enough. Once the new Alphie located Omega, it was to seize control of the host computer's operating system immediately, halt all other processes and transmit a copy of Omega to the Whitney College computer. If Alphie's original statement about Omega was right, it would not tolerate another copy of itself running anywhere. We would have a bargaining position with the all-powerful program.

"But wouldn't it be better simply to destroy the copy out there?"

"Won't work," declared Gilbert. He strode nervously to a window where rain had begun to spatter. Then he paced back.

"Omega probably has a back-up copy of itself to be activated only when the original fails to report its existence to a triggering program at preset times. Anyway, Omega is no longer in Montana..."

Gilbert sighed and slumped down in a swivel chair. He looked pale and tired.

"He's been up all night," explained Reggie, "The new Alphie is out there right now, going from site to site. As soon as it finds Omega, Alphie will transmit the code here. It's the only computer around with enough memory to hold the copy of Omega. A real, cybernetic prison."

Reggie grinned at his own wit but Gilbert did not seem amused. He trundled his swivel chair over to a terminal, glanced at the screen and began to type some commands.

"Not in Oregon either. Hmmm. I wonder..."

Reggie was at the window.

"Hey, there's a car out front with two guys in it!"

"Lock the door!"

I ran to the window. The rain had become heavy and thunder rumbled through the sky as two men in gray suits got out of a car and mounted the steps of the Physics Building. Had they been following Gilbert? Were they agents of Omega?

"Hmm. Omega might have gone North to Canada. Let's try the University of British Columbia."

Two heads appeared outside the computer room. Suddenly the door crashed open and the men were inside the room. One of them drew a gun, waving Reggie and me into a corner. The other man rushed to Gilbert's terminal.

"You. Stay put and type this message: '57 and 48 are here.' Go on. Move it!"

Gilbert typed. The room was silent. Then, one by one, the disk memory cabinets whirred to life. It

was not until Gilbert spoke, his voice cracking with fear, that I began to understand the sense of a powerful presence filling the room:

"Looks like Omega is paying us a visit."

The idea of Omega itslef in that very room sent shivers down my spine. It almost made me miss the deliberately melodramatic tone in Gilbert's voice.

"Must have come in through the patch panel."

He drew out the last two words, pronouncing them with unusual care.

"Shut up!" The man clipped Gilbert across the back of his head.

The patch panel was a gray box on the wall directly beside us. Inside the box were all the lines connecting the mainframe to terminals across the campus – and to the outside world.

Before I knew it, Reggie had jumped to the panel and flipped open the door. He began to claw at the wires with his hands, tearing his flesh, scrabbling desperately.

"Hey, stop that."

A gun cracked and chips of plaster blew away from the wall beside Reggie's nose. He dropped to the floor, cowering. Above him, loose wires splayed out of the patch panel in all directions, an electrician's nightmare.

"You're too late!" Gilbert laughed bitterly. "There's no way Omega can get back out."

"Reconnect those wires or I'll kill you."

The man approached me. His eyes were remote, uncaring.

"With the best will in the world," I said, "there's no way I can reconnect..."

The world went dark.

When I woke up again, I found Gilbert and Reggie staring at me. They were laughing.

"Too bad you missed the ending," said Gilbert.

When Omega's agents realized they would be unable to reconnect Omega, they had consulted the operator's console. Then they had fled without a word, as though something was about to happen.

The screen on the console told the story:

TO 57 AND 48:
RECONNECT PORTS AT ONCE.
IF YOU DO NOT ACCOMPLISH
THIS IN TEN MINUTES YOU
WILL BE TERMINATED.

According to Gilbert and Reggie, Omega had developed a some-what more engaging personality since the departure of its agents. It had explained that the business with the two men had been a mistake. Had Gilbert ever wished for a million dollars?

"What did you do?"

"Well, considering that those boys were originally sent to do a number on us under Omega's direct supervision, I was not in a terribly kind mood. Can you imagine what Omega was about to do? It could have used the men to interrogate us directly, to beat whatever out of us–or into us–that it wanted.

"Omega's still in there. I made a copy on magnetic tape in case it gets any fancy ideas about self-destructing. I imagine there's already a second Omega in the great outer world. It's time we talked to it."

This turned out to be no more complicated than sending a message to Omega Software through its mailnet address. We explained that we had the original Omega and enough working copies of the code to disrupt Omega Software permanently if we wished. Our message was acknowledged but not replied to.

Two days later, Gilbert got a telephone call from a man with a Greek accent who identified himself as Mr. Omega. His company, said the man, was prepared to be reasonable. In exchange for "certain software protection services" which only we could provide, Omega Software would pay him approximately half its profits. Gilbert had some additional conditions to propose. The new Omega would have to mind its manners. As part of our protection services we would, naturally, arrange that copies of Omega would be released from a great variety of sites (not to mention an army of ruthless Alphie's) should push come to shove.

The sum offered by Omega was enormous. Gilbert and I split the money. After amassing one of the world's largest collections of new and antique motorcycles, Gilbert has lapsed into what can only be called riotous living. Not that I'm a puritan. I've paid off the house, bought a new car, and had a little fun myself. But what to do with the appalling sum at my disposal?

It frightens me to think that a collection of seven million instructions can not only operate with perfect logic in the human sphere, but can yearn for survival and boil with hatred: There are sometimes people who watch me. My wordly circumstances are increasingly hemmed in by small but significant events.

My hand is forced. How ironic that I must give up the peace that I could not afford until recently! There is not much time before the new Omega gains some permanent advantage that we cannot now foresee.

Even as I write this, the AZ Institute is taking shape. Aided by the finest minds that money can buy, we will analyze Alphie down to the last instruction. We will uncover the lost principles of his operation. And what of the old Omega? It may not be necessary to analyze any member of the staff.

BULLETIN BOARD

Software and publications available from individuals and small companies.

Algorithm

BULLETIN BOARD ITEMS ARE FREE!
Your first posting on our bulletin board is free. Subsequent postings will cost $20.00 (US) each. Algorithm accepts no responsibility for the veracity of claims made about products or services offered herein.

What is the APL Programming Language?

APL is a quick and efficient tool for exploratory numerical analyses of Algorithm puzzles/articles.

Call the BBS\APL 301-384-3672 3/12/2400 baud, 24 hrs. Many downloadable files (IBM/DOS, not Apple), including 2 free APL interpreters.

Quelques fiches et menus en français.

J. of B.A.S.R.A.

Journal of the British-American Scientific Research Association
Specializing in avant garde science and speculations in science.
A forum for ideas in the sciences.
For information write to: **BASRA**
13 Durwood Place
Madison, NJ 07940

Searching for REAL fast algorithms?
Only 10 secs for any Mandelbrot-graph!
Speedy 3D-Life. 2D-Life: 90 generations/sec
Evolution. Infection. Many others. Sources included.
Send US $20 for 3.5" disk (Atari ST color/mono)
or US $5 for paper listing of algorithms and sources.

SpeedGraph Tiistilankuja 1 E 50, SF-02230
Espoo, FINLAND

New Release
WHAM-DUZLE Graphical Diversion
Experimental Design Kit with source code.
Requires 5.25" PC Compatible with MicroSoft or GWBASIC
Color Monitor optional.
Non-Objective Minimalist Art by rapid tessellation of extended character set.
Send $30 plus tax check or money order:
Christopher Computer, 28 Anderson St.,
Boston, MA 02114-3648

If you have a Commodore 64 or 128

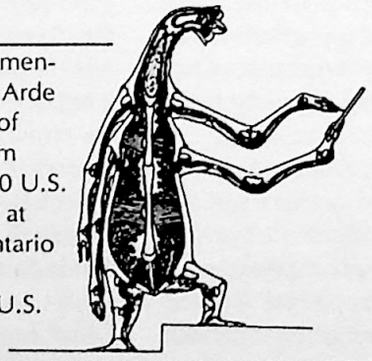
The world of mathematical artistry is within your reach too!
Mandelbrot Set - full control, colour, zoom, high speed.
Cellular Automata - set rule, seed, edge etc.; or auto. Warp speed
Darwin - evolution on-screen in minutes! Scientific Am. May 89
3-D Projection - view math functions & rotate 3-D objects.
Alphametic - high-speed on-screen solution of addition puzzles.
Magic Monkeys - animated, entertaining arithmetic for kids.
$10 each, or $30 for all: Ian Adam 4425 West 12th Ave.
Vancouver, BC, Canada V6R 2R3

Expert Systems on your PC

IBIS by
Intelligence Manufacturing, Co.
Inexpensive, Easy, and Powerful
Educational Version-Shareware $40.00
Development System $250.00
Contact: **D.E.M. Software**
P.O. Box 456 Woodbine NJ
08270 USA (609) 861-5181

YENDRED LIVES!

Follow Yendred, the Two-dimensional being, as he explores Arde and the Planiverse. Copies of *The Planiverse* available from A.K. Dewdney. Send $10.00 U.S. (or equivalent) to the author at 42 Askin Street, London, Ontario CANADA N6C 1E4
Overseas orders add $2.00 U.S. for shipping.



FREE!FREE!FREE!FREE!FREE!FREE!FREE!FREE!FREE!

Unleash the real power of computer simulation!
COMPUTER SIMULATION AND RECREATION BULLETIN BOARD
(Dedicated with passion to the public domain)
Want to help unleash popular fascination and spur unfettered development of Artificial Life, Biomorphs, Cellular Automata, Mandelbrot, Neural Nets, and Simulations of Natural Processes? Your talents, abilities, and know-how are needed. Link up with others who want to make a prestigious bulletin board dedicated to making these ends a reality.
Send for free "Draft of Proposed Principles (for discussion)" to:
CSRBB, P.O. Box 20714, Seattle, WA 98102

FREE!FREE!FREE!FREE!FREE!FREE!FREE!FREE!FREE!

EXCEL TEMPLATES AND ORIGINAL MACINTOSH ARTWORK

- LOTTO ANALYSIS
- WORLD DYNAMICS
- COVERSATION FACTORS
- PORTRAITS VOL 1
- PORTRAITS VOL 2

SEND $5.00 PER DISK TO COVER POSTAGE AND HANDLING TO:

RONALD A. OLANSEN
MACINTOSH ROAD, LYME, CT. 06371

BUY A BIT AT A TIME!

Shop at the world's only computer museum gift shop! The Computer Museum Store -- a one-of-a-kind shop featuring an imaginative selection of high-tech posters, games, jewelry, and books. Call or write for your free catalog.

The Computer Museum

300 Congress Street
Museum Wharf
Boston, MA 02210
(617) 426-2800 ext. 307

Whether or not you have or even want the book, you may like to have the ms-dos basica programs in it:

Lessons in Edgeworth's Box — graphics of General Equilibrium
The Hyperbox — m goods and n traders
LP Simplex Algorithm — specially for class excerise
Power Algorithm for Minimum Paths — PROD=SUM, SUM=MIN
Von Neumann's Economic Model — the Stretch algorithm
Two Person Zero-sum Matrix Games — Minimaz
Stochastic Matrix Powers — contraction map graphics
Shao Yung's Program ?! & c... — ...And a few additional items, some related to Kee Dewdney's articles in *Scientific American*.

Logic of Choice and Economic Theory

Oxford: Clarendon Press, 1987 — Review in Economic Journal, March 1989
S. N. Afriat — 452 Roxborough Ave, Ottawa K1M 0L2 — 5 1/4" OR 3 1/2", US $15

Fractal Explorer®

For IBM® PCs and Compatible
Only one fractal program offers:

• Almost unlimited zooming capability (10^{72}) • Full screen and quadrant displays • High-speed image generation - up to 100 times faster than the competition • Edge detection algorithm • Professionally designed user interface • Support for many high-res VGA cards (up to 1024x768) in both 16 and 256 color modes

Send $29.95 US (check or money order) to:
Andromeda Research, Inc., 6441 Enterprise Lane, Madison, WI 53719 VISA or Mastercard accepted.

Have you ever *seen* a function of a complex variable? Would like to really understand differential operators like *grad*, *div*, or *curl*? How about a peek into the 4th dimension?

• • •

Ask about our latest software for PC or Macintosh

Lascaux Graphics 3220 Steuben Ave.
Bronx, NY 10467 USA (212) 654-7429

*** 'I Owe Plenty' ---'Mark III' ***

Gives Amortization Schedules of Loans & Extra Payment/ Other Investment Advice. For PC Compaibles. Most versatile and crystal clear. Easily shows *huge saving* usually possible.

Pleased or money back, for only $18.00 (in Hawaii add 4%)

Joe M. Jeffers
92-1041 Makakilo Drive
#83, Ewa Beach, HI USA 96707
With bonus, latest TOPO set of Seven Games.
(State diskette size wanted)

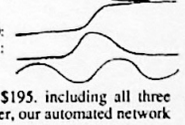
Loglan 1: **A Logical Language**
4th ed., 599 pp, $24. postpaid

A computer-intelligible scientific language, able to incorporate the vocabulary of any science. Teaching software, cassettes, parsers, dictionaries, workshops available.

The Loglan Institute
1701 NE 75th St., Gainesville, FL 32601
(904) 371-9574

New! BrainMaker V2.0
The #1 selling Neural Network.
Over 7,000 copies sold. Here's why:

Best documentation: 255 pg. *Introduction to Neural Networks*. 332 pg. Users Guide. 80 pg. Application Guide. Our documentation includes everything you need to know to build networks and applications now.
Easiest to use: Pull down menus and dialog boxes. Supports mouse, color. NetMaker automatic network generation program helps you build networks in seconds.
Fastest:
128,000 cps on 286/8:
250,000 cps on 285/20:
500,000 cps on 386/33:
3 Million cps on C25:
4 Billion cps on N64:
• BrainMaker is $195. including all three books and Netmaker, our automated network generation program.



"BrainMaker is the most fascinating computer software I've ever seen. This is hotter than hot."
John Dvorak
PC Magazine

Call or write for a **Free Catalogue!**
California Scientific Software
160 Montecito #E Sierra Madre, CA 91024
(818) 355-1094

Cellular Automata

for IBM PC. Mathematical pseudo-life grows, lives, and dies on your screen. Three programs offer speed, versatility, exceptional ease of use. Sophisticated user interface. Source code available. Demonstrated at CA conference at MIT. Send self-addressed stamped envelope for samples and full details, to:
Charles Platt (software), P. O. Box 556
Chelsea Station, New York, NY 10113

LETTERFORMS & ILLUSION

Scott Kim and Robin Fe Samelson

This set of computer programs is a collection of entertaining images, activities, puzzles and fonts that invites readers to explore for themselves the fascinating and challenging world of visual perception.

Designed for use with MacPaint™, the box consists of a completely self-contained 3½" disk and a 48-page book which guides the reader through the programs and projects. The simple programs of LETTERFORMS & ILLUSION will prove personally rewarding to the computer enthusiast or general reader with an interest in geometry, art, and design. ISBN 0-7167-2043-4, $39.95

W. H. Freeman and Company
41 Madison Avenue New York, NY 10010

COPY-WRITE
THIS PAGE US A LETTER

Moving? Have a product or service to distribute? Wish a friend to subscribe? Want to suggest a topic? Fill in the appropriate space below and mail the copy to us at:

Algorithm
P.O. Box 29237 Westmount Postal Outlet
785 Wonderland Road South
London, Ontario, Canada N6K 1M6

Name .....

Old address	New address

effective date: .....

If you're moving, let us know by filling out the space on the left. This way you can ensure an uninterrupted flow of fascinating projects and engaging articles.

Perhaps you know someone who would like to subscribe to Algorithm, the Personal Programming Newsletter. If so, copy this page and fill in the space on the right.

PLEASE send me 6 issues of Algorithm for the introductory price of **$29.95**

NAME:_____

STREET:_____

CITY:_____

STATE:_____

COUNTRY:_____ ZIP:_____

PLEASE SEND PAYMENT TO:
Algorithm P.O. Box 29237, Westmount Postal Outlet, 785 Wonderland Rd., London, Ontario, Canada N6K 1M6
(Cheque or money order. No C.O.D.)
Can. Funds $35.95 • Overseas add $4.00 Shipping.

If you have a product or service you'd like to make available to our readers, fill out the space to the left with a rough sketch of your ad and we will set it in type. First appearance: FREE! Subsequent appearances: $20.00/issue.

Is there a topic you'd like to see us cover? Fill out the details in the space below:

Dear Editor: _____

_____

_____

_____

_____

_____

Name: _____ Phone: _____

Address: _____ Date: _____